Scale Invariability in Receptive Field /Stitcher

- TridentNet与Stitcher表格总结

TridentNet与Stitcher表格总结

- TridentNet三叉戟结构，通过三支不同大小的receptive field来适应不同尺度大小的物体，分支参数共享，从而提升性能；
- Stitcher：通过在每一个batch中设置一个阈值，将大中型物体转化为中小型，弥补了batch内小物体的数量，使小物体的分布更加均匀，达到数据增广的作用，提升了性能。
- 对比表格总结链接：
- 论文对比总结\TridentNet与Stitcher表格总结.html

- TridentNet：Scale aware Trident Networks for Object Detection　CVPR.2019

TridentNet

- 论文中作者首先设计了一个实验来验证感受野对不同尺度物体检测的影响。基于实验结果，提出了TridentNet（三叉戟网络），旨在生成具有统一表示能力的尺度特征映射。
- 本文构造了一个并行的多分支体系结构，其中每个分支共享相同的转换参数，但是具有不同的感受野。
- 然后，提出了一种尺度感知的训练方案，在训练过程中为每个分支采样指定比例的对象，并共享三个分支上参数进行学习。

传统方法的不足

- 对尺度变化的处理。目标尺度的大范围变化在目标检测中是很常见的，这种尺度的不确定性，尤其是过大或者过小的目标，对检测器是一个很大的挑战。

- 单阶段网络的代表算法有YOLO系列，SSD等，核心思想是原图上划分若干区域，直接进行分类和回归，优点是速度很快，但是准确率相对不高；

- 双阶段的方法核心思想是首先粗略提取出候选区域，然后针对每个候选区域进一步细化，进行分类和位置回归操作。优点是能够提升准确率，但由于把整个过程分为两大步，因此在速度上有所欠缺。

Motivation

- 考虑对于一个detector本身而言，backbone有network depth（structure），downsample rate和receptive field会影响性能。

- 网络越深（或叫表示能力更强）结果会越好，下采样次数过多对于小物体有负面影响。但是没有工作，单独分离出receptive field，保持其他变量不变，来验证它对detector性能的影响。
- 因为相机导致图像中同类物体出现不均一尺度。卷积缺少尺度不变性。从scale variation出发，针对不同大小的物体，感受野应该是不同的。

Motivation

- 随着感受野的增大，小目标的检测准确性也开始下降，但是大目标的检测准确性开始上升，证明不同尺度物体的检测性能和dilation rate正相关。

- 说明传统的Faster-RCNN中的感受野还是不够大，更大的receptive field对于大物体性能会更好，更小的receptive field对于小物体更加友好。

Backbone	Dilation	AP	AP_s	AP_m	AP_l
ResNet-50	1	0.332	**0.174**	0.384	0.464
	2	0.342	0.168	**0.386**	0.486
	3	0.341	0.162	0.383	**0.492**
ResNet-101	1	0.379	**0.200**	**0.430**	0.528
	2	0.380	0.191	0.427	**0.538**
	3	0.371	0.181	0.410	**0.538**

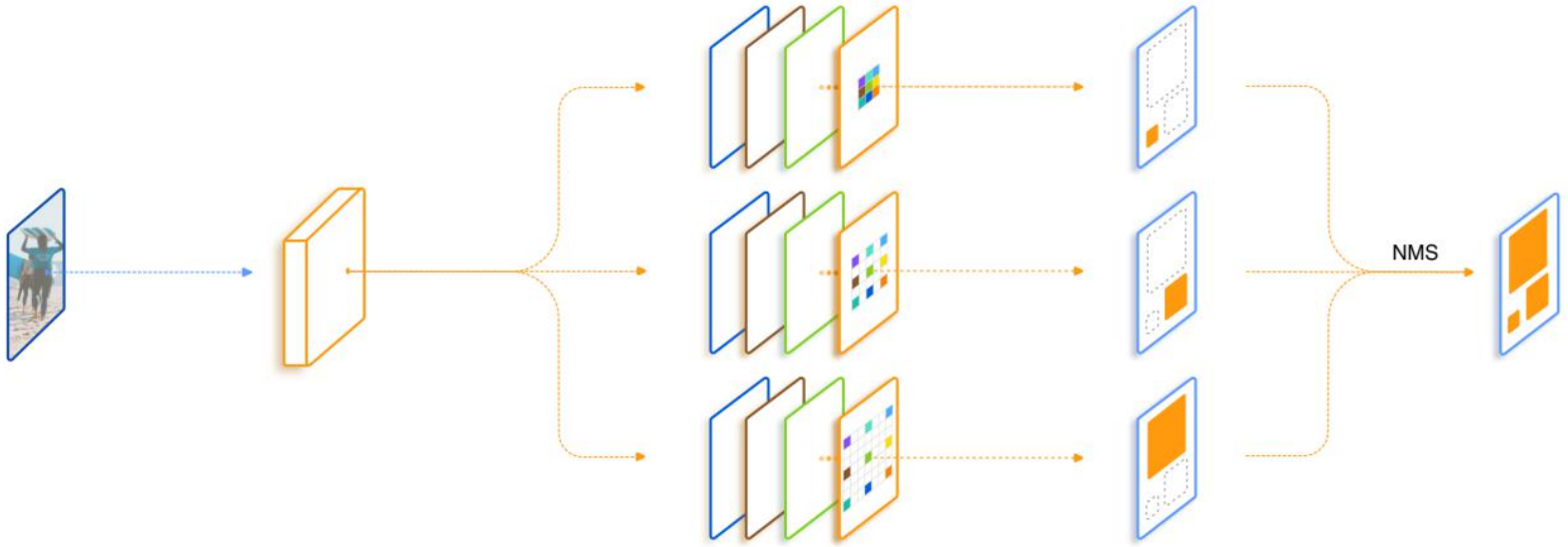
消融实验1：验证不同分支的作用

- 在各单独的branch上进行训练与利用三个分支共享参数进行训练的性能
- 测试中划分小、中、大目标的尺度范围[(0,32),(32,96),(96,+∞)]
- 3 branches的性能最好；同时branch-2的性能是单独使用一个branch中性能最高的。

Method	Branch No.	AP	AP_{50}	AP_s	AP_m	AP_l
Baseline	-	37.9	58.8	20.0	43.0	52.8
TridentNet	Branch-1	31.5	53.9	22.0	43.3	29.9
	Branch-2	37.8	58.4	18.0	45.3	53.4
	Branch-3	31.9	48.8	7.1	37.9	**56.1**
	3 Branches	**40.6**	**61.8**	**23.0**	**45.5**	55.9

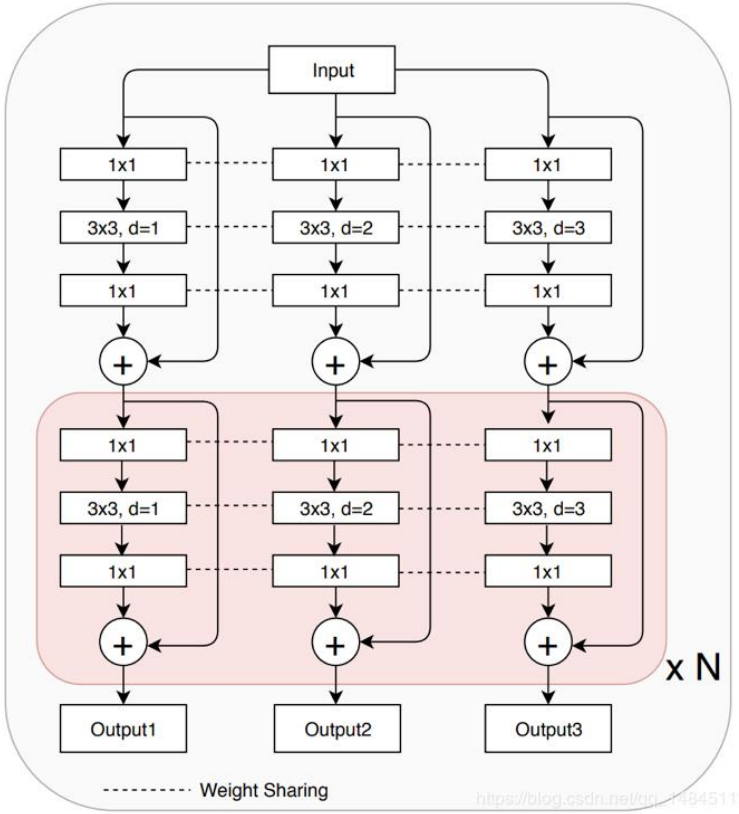
主要框架

- TridentNet 模块主要包括3个完全一样的分支，唯一不同的只是膨胀卷积的膨胀率。从上到下，膨胀率分别为1,2,3，分别检测小，中，大的目标。
- 作者令三个分支参数共享，使不同分支的网络结构完全相同，仅仅Dilation系数这个超参数不同。



主要框架

- 与SNIP相似，仅选择尺度在每个分支的相应有效范围[li, ui]内的proposals和GT进行训练。Scale aware的range为 [(0, 90), (30, 160), (90, +∞)]
- 在测试时，只用中间的分支进行推理，因为其学习了不同scale的目标；然后对结果进行NMS后处理，最后输出预测的目标信息。
- 用一个主分支就可以接近最优效果，只用一个分支就能得到（接近）最后效果且耗费的资源更少。当然这样做会带来一些精度损失，大概在0.5-1个mAP，但这样的好处在于不会引入额外的参数，不会增加额外的计算量。



实验结果

- Multi-branch：a，b的对比说明受益于不同的感受野的多分支结构能涨1.1 mAP。
- Scale-aware：b，d的对比可说明Scale-aware能提升小目标0.8 mAP的性能，但大目标的性能降低0.9 mAP。

Backbone	Method	Multi-branch	Weight-sharing	Scale-aware	AP	AP_{50}	AP_s	AP_m	AP_l
ResNet-101	(a) Baseline	-	-	-	37.9	58.8	20.0	43.0	52.8
	(b) Multi-branch	✓			39.0	59.7	20.6	43.5	55.1
	(c) TridentNet w/o scale-aware	✓	✓		40.3	61.1	21.8	44.7	**56.7**
	(d) TridentNet w/o sharing	✓		✓	39.3	60.4	21.4	43.8	54.2
	(e) TridentNet	✓	✓	✓	**40.6**	**61.8**	**23.0**	**45.5**	55.9
ResNet-101-Deformable	(a) Baseline	-	-	-	39.9	61.3	21.6	45.0	55.6
	(b) Multi-branch	✓			40.5	61.5	21.9	45.3	56.8
	(c) TridentNet w/o scale-aware	✓	✓		41.4	62.8	23.4	45.9	**57.4**
	(d) TridentNet w/o sharing	✓		✓	40.3	61.6	22.9	45.0	55.0
	(e) TridentNet	✓	✓	✓	**41.8**	**62.9**	**23.6**	**46.8**	57.1

实验结果

- Weight-sharing：c，e说明在所有分支共享相同的参数的帮助下，可减少参数数量并提高了检测器的性能。

- 作者解释性能提升是因为：共享的参数在所有规模的对象上都经过全面训练，从而减轻了对某一个scale的训练中过度拟合的问题。

关于scale aware的range大小的设定实验

- 证明最好的setting其实是没有scale aware。

- 可以理解成训练的时候一个样本在不同的receptive field下训练起到了augmentation的作用。

Scale-aware Ranges	AP	AP_{50}	AP_s	AP_m	AP_l
(a) Baseline	37.9	58.8	20.0	43.0	52.8
(b) $[0, 90], [30, 160], [90, \infty]$	37.8	58.4	18.0	**45.3**	53.4
(c) $[0, 90], [0, \infty], [90, \infty]$	39.3	60.1	19.1	44.6	56.4
(d) $[0, \infty], [0, \infty], [0, \infty]$	**40.0**	**61.1**	**20.9**	44.3	**56.6**

Branch数量实验

- 当三叉戟块的数量超过10个时，TridentNet的性能将变得趋于稳定。
- 当数量超过15个时，性能开始下降。

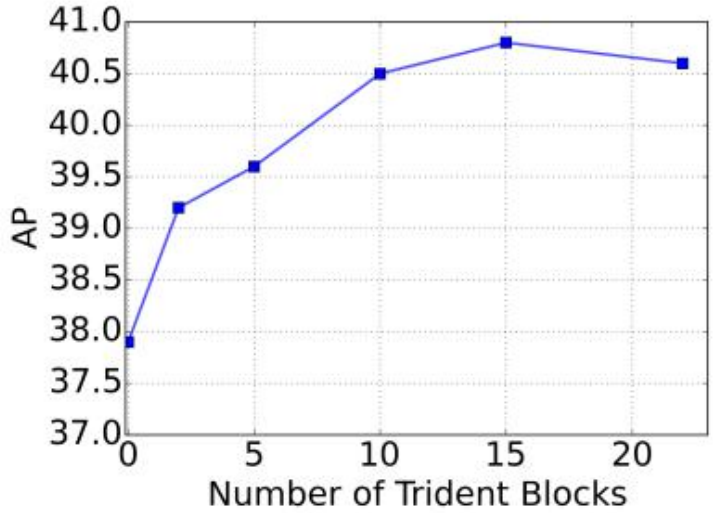


Figure 4: Results on the COCO *minival* set using different number of trident blocks on ResNet-101.

启发性或普适性

- 1.没有scale aware的setting的三叉戟网络，一个样本在不同的receptive field下起到了数据增广的作用，通过共享权重参数带来对各种尺度适应性，这些参数在所有规模的对象上都经过全面训练，从而减轻了对规模的训练中过度拟合的问题。

- 2.权值共享有以下几个优点：可以不增加额外的参数量。和论文的出发点一致，即不同尺度的物体应该以同样的表征能力进行统一的转化。在不同的感受野下，可以对不同尺度范围训练相同的参数。

总结

- TridentNet = 对同一物体使用不同大小的感受野来实现数据增广 + 共享权重参数带来对各种尺度适应性

我的观点

- 文中的采样多分支并行在同一个特征图上处理，而常规的多尺度分支预测是在不同特征图上进行。
- 最大亮点用dilate conv来变换scale，从而检测多尺度。从结构上处理了不同的scale，让有效的参数去学习更有语义的信息，而不是去“记忆”scale。
- 本文出发点是实验验证了增大感受野可以涨点，提高了大物体的识别效果。但并没有考虑对极小物体的检测效果的优化，Scale-aware的应该是能起到一定的作用的，但在小物体的检测上还有优化的可能。

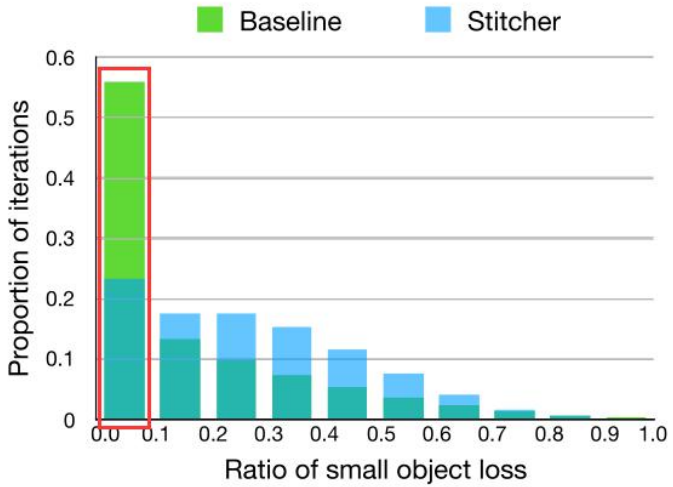
- Stitcher：Feedback-driven Data Provider for Object Detection CVPR.2019

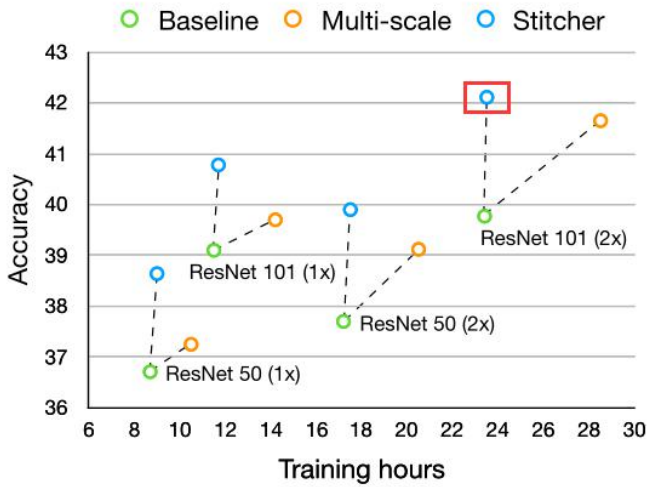
Stitcher

- 通过对每张图片设置一个小目标的loss阈值，低于阈值时把batch内每4张图都缩小到同样大小，之后拼成一张与正常普通同样大小的图作为训练。
- 本质是把大物体和中物体缩小成中物体和小物体，来均衡不同Scale物体在训练过程中的分布。

传统方法的不足

- 有超过50% iterations中，小物体所产生的loss都非常低（不到总loss的0.1）。
- 这说明在模型训练过程中，小物体提供给网络的监督是不足的。
- 数据集中小物体分布不均匀 --> 训练中小物体学习不充分（Loss不足） --> 训练完的模型小物体精度差。





Motivation

- 在最常用的baseline: Faster R-CNN + ResNet 50-FPN (1x) 的结果中
- AP small 比 AP large 低了两倍还多。
- 小物体到底出了什么问题，以及怎样解决这样的问题。

AP	AP small	AP mid	AP large
36.7 %	21.1 %	39.9 %	48.1 %

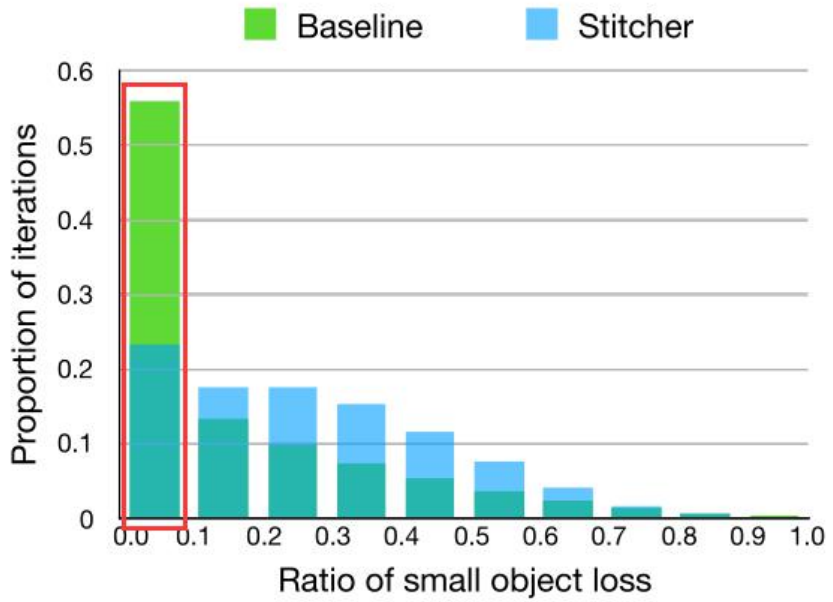
Motivation

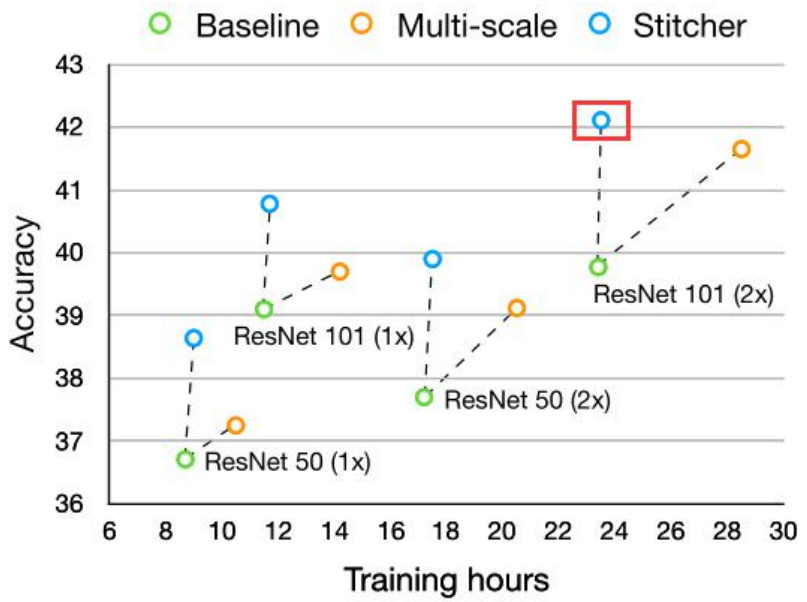
- 统计了小物体在数据集中的分布，发现训练集中小物体的数量并不少。
- 在所有boxes中，有41.4 %是小物体，是这三类物体之中最多的；但小物体的分布却非常不均匀，只有52.3%的图片中包含了小物体，而中物体和大物体分布都相对均匀。
- 小物体数量很多、但分布非常不均匀，有接近50%的图片中都没有小物体。

	Small	Mid	Large
Ratio of total boxes (%)	**41.4**	34.3	24.3
Ratio of images included (%)	**52.3**	70.7	83.0

Motivation

- 对于Faster R-CNN baseline，在超过50%的迭代中，小物体所产生的loss贡献非常低（不到总loss的0.1）。采用Stitcher时，损失分配变得平衡。
- 这说明在模型训练过程中，网络中对小物体的监督是不足的。



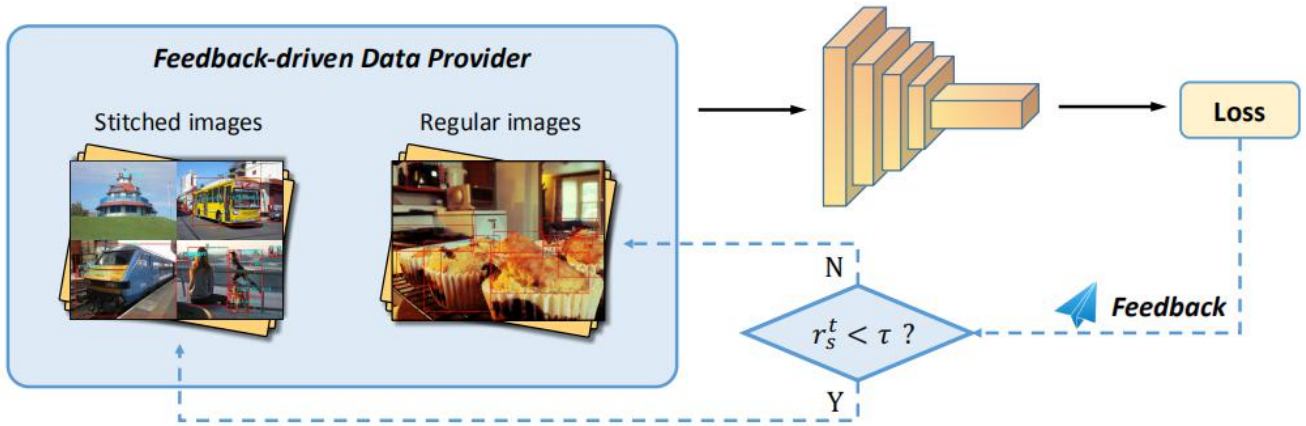


Motivation

- 1. 数据集中小物体分布不均匀 。
- 2. 导致训练中小物体学习不充分（某些图片上有多个小物体，而47.7%的图片则没有小物体，导致其在学习时Loss不足）。
- 3. 某些batch中没有学习到小物体，导致训练完的模型小物体精度差。
- 4. 采用Stitcher的方法，来加强对小物体的监督。

主要框架

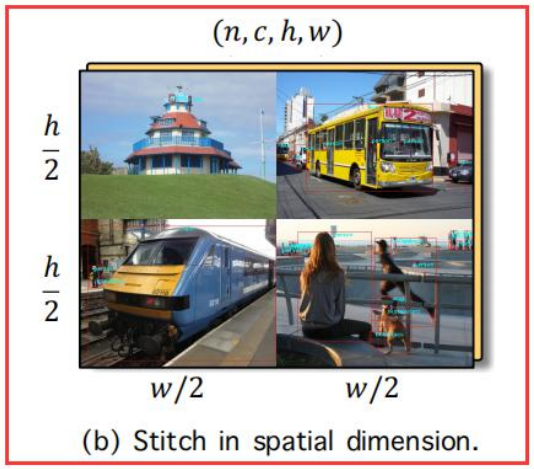
- 通过用小物体的loss作为反馈信号，将4张图都缩小到同样大小，之后拼成一张与正常普通同样大小的图作为训练，从而制造更多的小物体，借助图像拼接可以减轻图像batch（充当最小训练实体）的比例失衡，弥补训练集中几乎有一半的图像不包含小物体的问题。
- 实质上是一种“缺啥补啥”的简单思路：如果上一个iteration中，小物体产生的loss不足（比例小于一个阈值），则下一个iteration就用拼接图；否则就用正常图片训练。



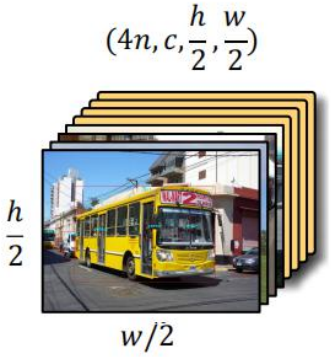
Stitcher

- 一批形状为（kn，c，h /√k，w /√k）的缝合图像，其中图像沿批次尺寸n连接。下图的为可视化设置k = 4
- 不仅可以在Spatial维度 (h,w)上进行拼接，还提供了一种在batch维度n上拼接的等价的实现方式。

(a) Regular Images.



(b) Stitch in spatial dimension.



(c) Stitch in batch dimension.

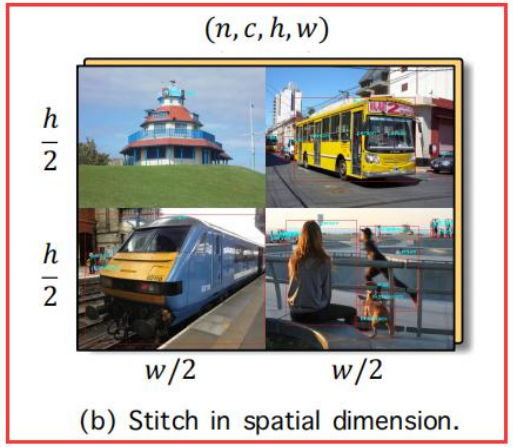
Dimension	k	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Spatial	4	38.6	60.5	41.8	24.4	41.9	49.3
Batch	2	38.3	60.2	41.7	22.9	41.3	49.5
	3	38.5	60.5	42.0	22.9	41.8	49.1
	4	38.6	60.6	42.1	23.4	41.5	50.3
	5	**38.7**	60.8	41.9	23.7	41.6	50.1
	6	38.6	60.7	42.1	23.5	41.5	49.5
	7	38.4	60.5	41.8	23.6	41.5	48.6
	8	38.3	60.6	41.6	24.3	41.3	49.0

Table 11: Concatenation dimensions

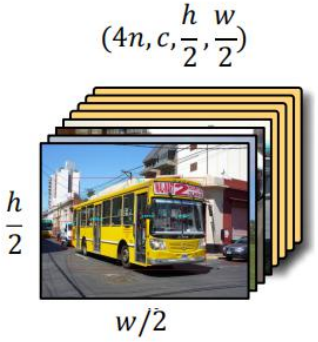
Yolov4的Mosaic

	Stitcher	YOLOv4-Mosaic
相同点	混合4张图片变为一张图，加入训练(数据增强)	
不同点	一次性能拼接k张图片	能调整四张图片为不同的大小

(a) Regular Images.



(b) Stitch in spatial dimension.



(c) Stitch in batch dimension.


aug_-319215602_0_-238783579.jpg


aug_-1271888501_0_-749611674.jpg


aug_1462167959_0_-1659206634.jpg

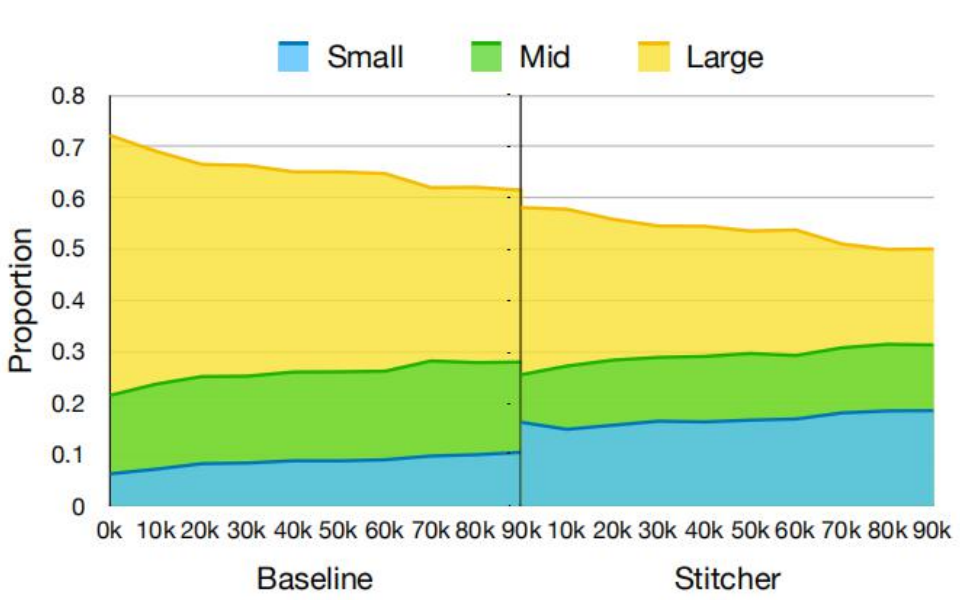

aug_1474493600_0_-45389312.jpg

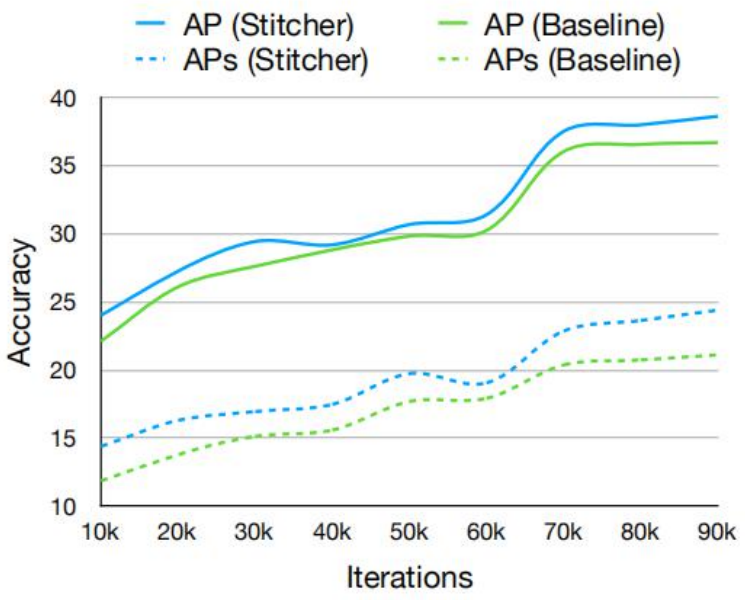

aug_1715045541_0_603913529.jpg


aug_1779424844_0_-589696888.jpg

实验

- 左图表明，使用Stitcher，各种规模的loss分布更加平衡，右图显示stitcher提高了精度。





实验

- 在同一backbone上，比Faster R-CNN提升2 mAP左右。
- 在小目标上的提升更为显著，约3.5 AP，也符合作者的motivation。

Table 2: Comparison with common baselines on Faster R-CNN

	period	backbone	AP	AP_s	AP_m	AP_l
Baseline	1×	Res-50-FPN	36.7	21.1	39.9	48.1
Stitcher			**38.6 (+1.9)**	**24.4 (+3.3)**	41.9 (+2.0)	49.3 (+1.2)
Baseline		Res-101-FPN	39.1	22.6	42.9	51.4
Stitcher			**40.8 (+1.7)**	**25.8 (+3.3)**	44.1 (+1.2)	51.9 (+0.5)
Baseline	2×	Res-50-FPN	37.7	21.6	40.6	49.6
Stitcher			**39.9 (+2.2)**	**25.1 (+3.5)**	43.1 (+2.5)	51.0 (+1.4)
Baseline		Res-101-FPN	39.8	22.9	43.3	52.6
Stitcher			**42.1 (+2.3)**	**26.9 (+4.0)**	45.5 (+2.2)	54.1 (+1.5)

实验

- 在同一backbone上，比RetinaNet提升2.2 mAP左右。

Table 3: Comparison with common baselines on RetinaNet

	period	backbone	AP	AP_s	AP_m	AP_l
Baseline	1×	Res-50-FPN	35.7	19.5	39.9	47.5
Stitcher			**37.8 (+2.1)**	**22.1 (+2.6)**	41.6 (+1.7)	48.9 (+1.4)
Baseline		Res-101-FPN	37.7	20.6	41.8	50.8
Stitcher			**39.9 (+2.2)**	**24.7 (+4.1)**	44.1 (+2.3)	51.8 (+1.0)
Baseline	2×	Res-50-FPN	36.8	20.2	40.0	49.7
Stitcher			**39.0 (+2.2)**	**23.4 (+3.2)**	42.9 (+2.9)	51.0 (+1.2)
Baseline		Res-101-FPN	38.8	21.1	42.1	52.4
Stitcher			**41.3 (+2.5)**	**25.4 (+4.3)**	45.1 (+3.0)	54.0 (+1.6)

实验

- 就准确度而言，Stitcher相对于多尺度训练的优势主要是从小尺度获得的。它们在检测大型物体方面具有大致相同的能力。
- 这样的对比证实了我们设计的目标，主要目的是通过图像拼接来检测小物体。

Table 4: Comparison with multi-scale training on Faster R-CNN

	period	backbone	hours	AP	AP_s	AP_m	AP_l
Multi-scale	1×	Res-50-FPN	10.5	37.2	21.6	40.3	48.6
Stitcher			**9.0**	**38.6 (+1.4)**	**24.4 (+2.8)**	41.9 (+1.6)	49.3 (+0.7)
Multi-scale		Res-101-FPN	14.2	39.7	23.6	43.3	51.3
Stitcher			11.7	**40.8 (+1.1)**	**25.8 (+2.2)**	44.1 (+0.8)	51.9 (+0.6)
Multi-scale	2×	Res-50-FPN	20.5	39.1	23.5	42.2	50.8
Stitcher			**17.5**	**39.9 (+0.8)**	**25.1 (+1.6)**	43.1 (+0.9)	51.0 (+0.2)
Multi-scale		Res-101-FPN	28.5	41.6	25.5	45.3	54.1
Stitcher			**23.5**	**42.1 (+0.5)**	**26.9 (+1.4)**	45.5 (+0.2)	54.1 (+0.0)

实验

- 在Res-50-C4 backbone上，Stitcher比SNIP、SNIPER提升0.6，0.7 mAP。
- 在Res-101-C4 backbone上， Stitcher比SNIP、SNIPER提升2.5，0.8mAP。

Table 5: Comparison with SNIP and SNIPER on Faster R-CNN

	backbone	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
SNIP	Res-50-C4	43.6	65.2	48.8	26.4	46.5	55.8
SNIPER		43.5	65.0	48.6	26.1	46.3	56.0
Stitcher		**44.2**	64.6	48.4	28.7	47.2	58.3
SNIP	Res-101-C4	44.4	66.2	49.9	27.3	47.4	56.9
SNIPER		46.1	67.0	51.6	29.6	48.9	58.1
Stitcher		**46.9**	67.5	51.4	30.9	50.5	60.9

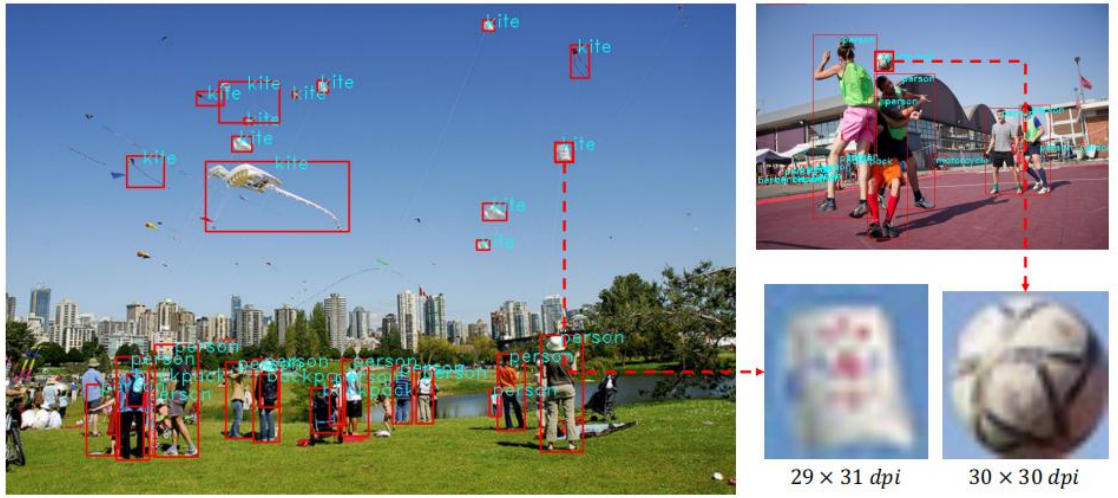
实验

- 如果一贯使用拼接图像，性能没有明显提升，表明Stitcher中随机采样是必不可少的，其可以当做是多尺度训练的特殊版本。

- 基于loss的feedback ratio比根据输入批次中小物体的数量计算比例input feedback性能高0.5 mAP，基于Regression loss feedback的性能最好，能达到38.6 mAP。

Table 10: Ablation study on selection paradigm

Selection Paradigm		AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
No feedback	All stitched	32.1	53.0	33.8	21.9	36.4	36.8
	All common	36.7	58.4	39.6	21.1	39.8	48.1
	Random sample	37.8	59.5	41.2	23.6	40.7	46.7
With feedback	Input feedback	38.1	60.0	41.2	23.1	41.3	49.1
	Classification feedback	38.5	60.6	41.9	23.9	41.6	48.8
	Regression feedback	**38.6**	60.5	41.8	24.4	41.9	49.3
	Total loss feedback	38.5	60.6	42.0	23.7	41.6	49.3



实验

- Stitcher中仅引入了一个超参数，即用于选择的阈值τ，将阈值设置为0.1时性能最高。
- 当阈值设置为0.4以下时，Stitcher的性能要优于“All stitched” 的baseline。否则，性能会迅速下降到“ All stitched ”。

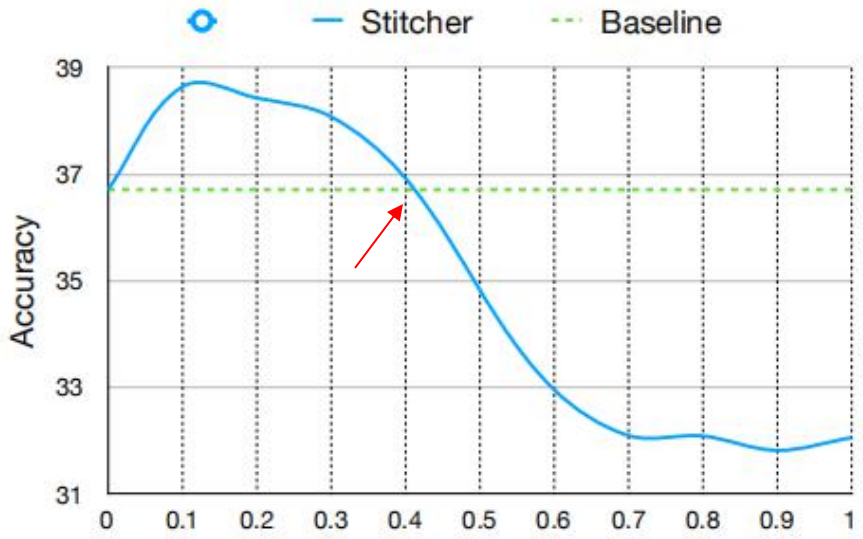


Fig. 8: Ablation study on threshold τ

启发性或普适性

- 1.在coco数据集中，小目标的数量最多，占比41.4%，但其只分布在52.3%的图片中，导致将近一半的图片中没有小物体，即有可能一张图片中有很多的小目标，而有的图片没有小目标。然而中、大尺寸的目标分布则较均匀。
- 2.可利用数据增广的方式，如在batch中增加小目标的数量，使物体scale分布更均匀，提升性能。
- 3.本文核心思想是利用每个iteration中的小物体loss的统计信息作为反馈，以自适应地确定下一个的输入选择(是否需要随机抽取不同大小的图片加入训练。

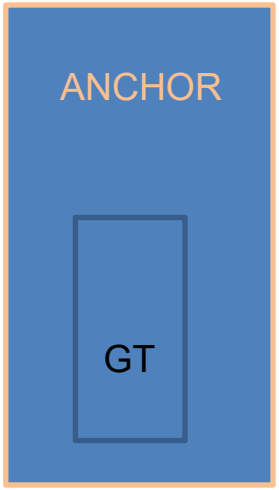
总结

- Stitcher = 利用单张图片中小目标loss的阈值，将大中目标转换中小目标，重新加入训练。(相当于数据增广)

我的观点

- 1.此文并没有根本地解决把小目标放大的问题，只是在小物体loss较小的情况下，把大中物体转化为中小物体，加入训练，相当于数据增广。

- 2.目前的目标检测都采用了Faster RCNN的Anchor思想，小物体有时候即使全部在Anchor内，也容易因为小面积导致IoU过低。



探究

- 1.是否可以引入三支决策的思想，
- 当小物体loss小时，采用stitcher；
- 当大物体loss小时，采用Crop或其他方法；
- 当其他情况时采取另一种方法。

三支决策

- **主要思想：** 将一个论域 划分为三个不相交的区域，并使用不同的策略进行处理。

$$f : U \longrightarrow \{POS, BND, NEG\}$$

- f： 可以看做一个函数或映射，一个对象通过 f 映射到{POS，BND，NEG}三个区域。
- U： 论域
- D： 条件集合

Chapter

- 存在问题

存在问题

- 1. TridentNet的table6中的scale aware问题。
- 2. Trident跑代码环境配置好像有点麻烦，还没有配好；问题解决了挺多的了，接下来稍微调下应该可以。
- 3. TridentNet文章的有的细节点作者好像没有说清楚，还需要往深点看。

存在问题

- 1. TridentNet的table6显示，没有scale过滤的是SOTA的，比scale-aware的方法性能更好，这好像表明此文的提出的三叉戟结果并不是一种性能先进的模型吧。
- 2. 假设主分支确实不应该要scale，那其他两个辅助分支应加上scale-aware的性能会不会好呢？毕竟不同分支感受野有差异，对于不同尺度的物体应该有不一样的强弱感受能力

Scale-aware Ranges	AP	AP_{50}	AP_s	AP_m	AP_l
(a) Baseline	37.9	58.8	20.0	43.0	52.8
(b) $[0, 90], [30, 160], [90, \infty]$	37.8	58.4	18.0	**45.3**	53.4
(c) $[0, 90], [0, \infty], [90, \infty]$	39.3	60.1	19.1	44.6	56.4
(d) $[0, \infty], [0, \infty], [0, \infty]$	**40.0**	**61.1**	**20.9**	44.3	**56.6**

我需要做的

- 1. 跑TridentNet源码，探究其参数共享与参数不同享的性能差异。

- 2. 对自己学过的关于目标检测中的不平衡问题做个小总结，着重关注于类别不平衡、尺度不平衡与空间不平衡。不平衡问题有：

- 类别不平衡：前景和背景不平衡、前景中不同类别输入包围框的个数不平衡；
- 尺度不平衡：输入图像和包围框的尺度不平衡，不同特征层对最终结果贡献不平衡；
- 空间不平衡：不同样本对回归损失的贡献不平衡、正样本IoU分布不平衡、目标在图像中的位置不平衡；
- 目标函数不平衡：不同任务（比如回归和分类）对全局损失的贡献不平衡。

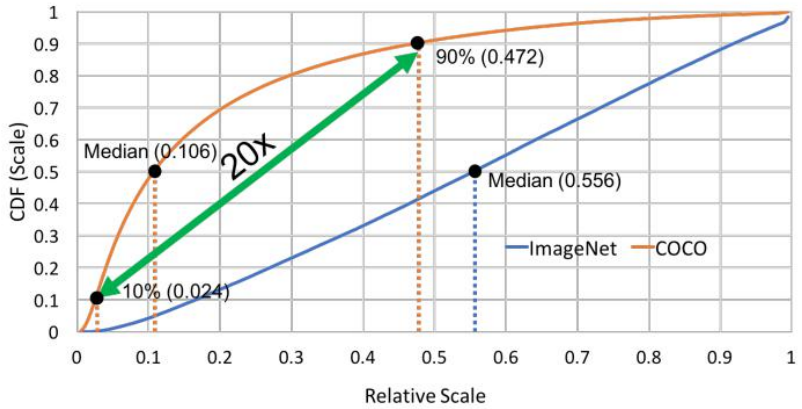
我需要做的

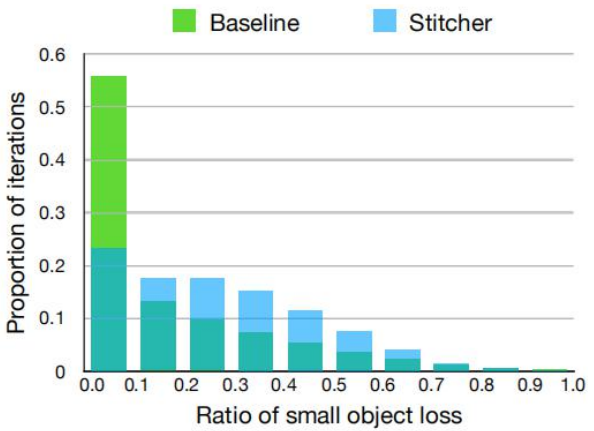
- 3. 可以多从SNIP、Stitcher、ATSS与yolo4中提出的coco数据集特征中发现问题，自己做小实验熟悉代码流程，进行改进。

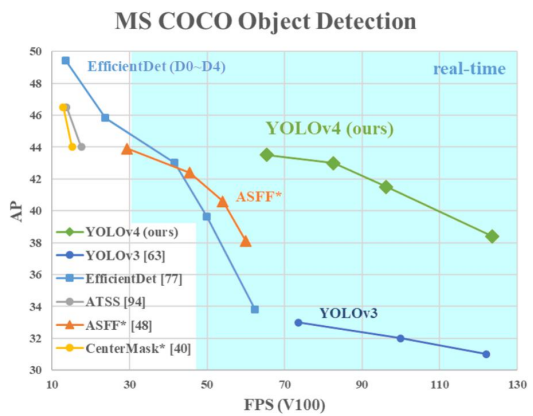
AP	AP small	AP mid	AP large
36.7 %	21.1 %	39.9 %	48.1 %

	Small	Mid	Large
Ratio of total boxes (%)	**41.4**	34.3	24.3
Ratio of images included (%)	**52.3**	70.7	83.0

Backbone	Dilation	AP	AP_s	AP_m	AP_l
ResNet-50	1	0.332	**0.174**	0.384	0.464
	2	0.342	0.168	**0.386**	0.486
	3	0.341	0.162	0.383	**0.492**
ResNet-101	1	0.379	**0.200**	**0.430**	0.528
	2	0.380	0.191	0.427	**0.538**
	3	0.371	0.181	0.410	**0.538**







我需要做的

- 4. yolo4中的Mosaic（马赛克）数据增强方法细节，yolo4内容较多，比较难，放在后面一点看。