# RetinaNet训练结果记录--新老版本比较

## 使用老版本的detectron

网络结构：retinaNet-101

预训练模型为R-101.pkl

### P2至P7，7个anchors比例和4个scale（ANCHOR\_SCALE为 4）

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为64；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 4.0, 5.0, 0.5, 0.3)；

图片大小：516\*576 （配置为512，576） Bathsize: 4

配置文件：configs\illbuild\retinanet\_R-101-FPN\_P7\_2\_R7.yaml

下面是训练和测试的结果：（retinanet\_R-101-FPN\_P7\_2\_R7.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 5万次 | 0.51🡪0.08 | 0.3284 | 0.3106 | 0.2341 |
| 10万次 | 0.51🡪0.06 | 0.3775 | 0.3776 | 0.2849 |
| 16万 | 0.51🡪0.05 | 0.3910 | 0.3910 | 0.2844 |
| 30万 | 0.51🡪0.01 | 0.3955 | 0.3926 | 0.2887 |

### P2至P7，6个anchors比例和4个scale（ANCHOR\_SCALE为 4）

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为64；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 4.0, 0.5, 0.3)

图片大小：516\*576 （配置为512，576） Bathsize: 4

配置文件：configs\illbuild\ retinanet\_R-101-FPN\_P7\_2\_R6.yaml

下面是训练和测试的结果：（retinanet\_R-101-FPN\_P7\_2\_R6.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 5万次 | 0.51🡪0.1 | 0.3158 | 0.3061 | 0.2251 |
| 10万次 | 0.51🡪0.1 | 0.3319 | 0.3294 | 0.2376 |
| 16万 | 0.51🡪0.1 | 0.3503 | 0.3451 | 0.2503 |
| 30万 | 0.51🡪0.08 | 0.3549 | 0.3501 | 0.2528 |

这两个实验的结果都不理想，查看检测的结果。有很多类别检测错误的情况，把很多工人检测为吊车。

【分析原因】

下面是训练集的数据统计：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.05~0.24 | 0.24~0.29 | 0.29~0.8 | 0.8~1.5 | 1.5~2.5 | 2.5~3.5 | 3.5~4.5 | 4.5~5.5 | 5.5以上 |
| 20.48~51.2 | 32 | 25 | 281 | 592 | 969 | 273 | 73 | 32 | 47 |
| 51.2~133.12 | 92 | 87 | 477 | 674 | 1078 | 274 | 85 | 54 | 63 |
| 133.12~215.01 | 29 | 44 | 261 | 221 | 206 | 70 | 72 | 10 | 1 |
| 215.04~296.96 |  | 9 | 153 | 115 | 54 | 107 | 25 | 1 |  |
| 296.96~378.88 |  |  | 84 | 77 | 41 | 11 |  |  |  |
| 378.88~460.8 |  |  | 45 | 109 | 17 |  |  |  |  |
| 460.8~542.72 |  |  |  | 45 |  |  |  |  |  |

下面是上面两个实验中在P2,P3,P4,P5,P6,P7层分类回归，这样每一层对应的anchors尺寸如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FPN  层 | Stride | 最小anchors的宽高对应网络输入图的尺寸的像素个数 | 最大anchors的宽高对应网络输入图的尺寸的像素个数 | 特征图大小  输入为512\*512 |
| P2 | 4 | 16 （4\*4） | 28 （7\*4） | 128\*128 |
| P3 | 8 | 32 （4\*8） | 56 （7\*8） | 64\*64 |
| P4 | 16 | 64 （4\*16） | 112 （7\*16） | 32\*32 |
| P5 | 32 | 128 （4\*32） | 224 （7\*32） | 16\*16 |
| P6 | 64 | 256 （4\*64） | 448 （7\*64） | 8\*8 |
| P7 | 128 | 512 （4\*128） |  | 4\*4 |

倒数第二层中256以上面积的1:5的比例不起作用（超过特征图尺寸了）， 320以上面积的1:3的比例也不起作用了，384以上面积的1:2的比例就不起作用了；

倒数第一层中除了1:1的比例，其他的比例都不起作用（超过特征图尺寸了）；

下面是训练集的数据统计，其中紫色的部分是上面两个实验没有覆盖到的：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0.05~0.24 | 0.24~0.29 | 0.29~0.8 | 0.8~1.5 | 1.5~2.5 | 2.5~3.5 | 3.5~4.5 | 4.5~5.5 | 5.5以上 |
| 20.48~51.2 | 32 | 25 | 281 | 592 | 969 | 273 | 73 | 32 | 47 |
| 51.2~133.12 | 92 | 87 | 477 | 674 | 1078 | 274 | 85 | 54 | 63 |
| 133.12~215.01 | 29 | 44 | 261 | 221 | 206 | 70 | 72 | 10 | 1 |
| 215.04~296.96 |  | 9 | 153 | 115 | 54 | 107 | 25 | 1 |  |
| 296.96~378.88 |  |  | 84 | 77 | 41 | 11 |  |  |  |
| 378.88~460.8 |  |  | 45 | 109 | 17 |  |  |  |  |
| 460.8~542.72 |  |  |  | 45 |  |  |  |  |  |

如果输入图片512\*512的，上面紫色的那部分始终难以覆盖，除非增大图片尺寸。

在不增大图片尺寸的前提下，只能让后面有效的anchors尽量多一些，比如将每个特征层ANCHOR\_SCALE从4修改为2，这样P6层就能多一组有效anchors，P7层也能多一些有效anchors。

如果修改ANCHOR\_SCALE为2，那么在P2,P3,P4,P5,P6,P7层分类回归，这样每一层对应的anchors尺寸如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FPN  层 | Stride | 最小anchors的宽高对应网络输入图的尺寸的像素个数 | 最大anchors的宽高对应网络输入图的尺寸的像素个数 | 特征图大小  输入为512\*51） |
| P2 | 4 | 8 （2\*4） | 20 （5\*4） | 128\*128 |
| P3 | 8 | 16 （2\*8） | 40 （5\*8） | 64\*64 |
| P4 | 16 | 32 （2\*16） | 80 （5\*16） | 32\*32 |
| P5 | 32 | 64 （2\*32） | 160 （5\*32） | 16\*16 |
| P6 | 64 | 128 （2\*64） | 320 （5\*64） | 8\*8 |
| P7 | 128 | 256 （2\*128） | 512 （5\*128） | 4\*4 |

倒数第二层中256以上面积的1:5的比例不起作用（超过特征图尺寸了）， 320以上面积的1:3的比例也不起作用了，384以上面积的1:2的比例就不起作用了；但是P6层也比之前能多一组有效anchors；

倒数第一层中256以上面积超过1:3的比例不起作用（超过特征图尺寸了），384以上面积得超过1:2的比例，其他的比例都不起作用（超过特征图尺寸了），即便这样但是P7层也比之前多一些有效anchors。

### P2至P7，6个anchors比例和4个scale（ANCHOR\_SCALE为 2）

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为64；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 2，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 4.0, 5.0,0.5, 0.3)

图片大小：516\*576 （配置为512，576）

下面是训练和测试的结果：（retinanet\_R-101-FPN\_P7\_2\_R7\_minScale\_2.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 2万次 | 0.94🡪0.08 |  | 0.1429 | 0.0817 |

### P2至P7，5个anchors比例和4个scale（ANCHOR\_SCALE为 4）

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为128；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 0.5, 0.2)

图片大小：768\*896 （配置为768，896）

2个GPU，batchsize总共为4。

下面是训练和测试的结果：（retinanet\_R-101-FPN\_1x.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 2.5万次 | 0.66🡪0.06 | 0.4688 | 0.4661 | 0.3810 |
| 4万次 | 0.66🡪0.04 | 0.4600 | 0.4582 | 0.3944 |
| 6万次 | 0.66🡪0.04 | 0.4715 | 0.4697 | 0.4047 |

效果还是不理想，再次调大图片，再次实验。

### P2至P7，6个anchors比例和4个scale（ANCHOR\_SCALE为4）

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为128；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 5.0, 0.5, 0.2)

图片大小：1024\*1024 （配置为768，896）

2个GPU，batchsize总共为2。

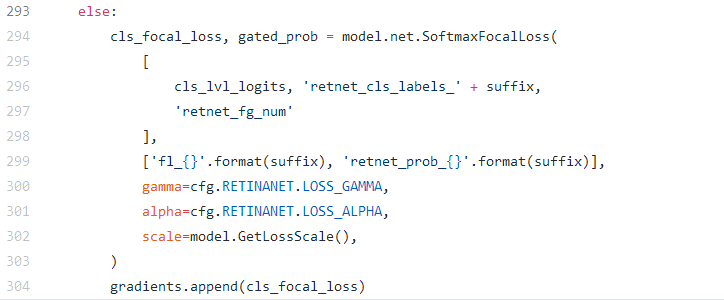
下面是训练和测试的结果：（retinanet\_R-101-FPN\_1x.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 1万次 | 0.66🡪0.1 | 0.1700 | 0.1599 | 0.1116 |

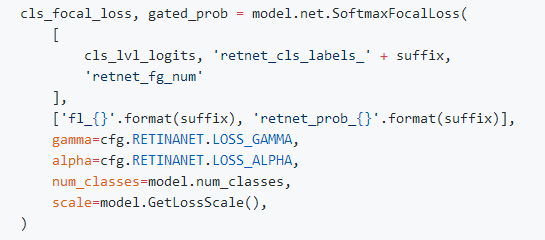
后来发现detectron的issue中有人也训练自己的数据，在某些类的AP值会为0，导致总mAP值也很低。这个问题是已经解决的问题：

<https://github.com/facebookresearch/Detectron/issues/93>

原因是[**Detectron/lib/modeling/retinanet\_heads.py**](https://github.com/facebookresearch/Detectron/blob/021685d42f7e8ac097e2bcf79fecb645f211378e/lib/modeling/retinanet_heads.py#L280-L304)中有如下的代码：



正确应该是：

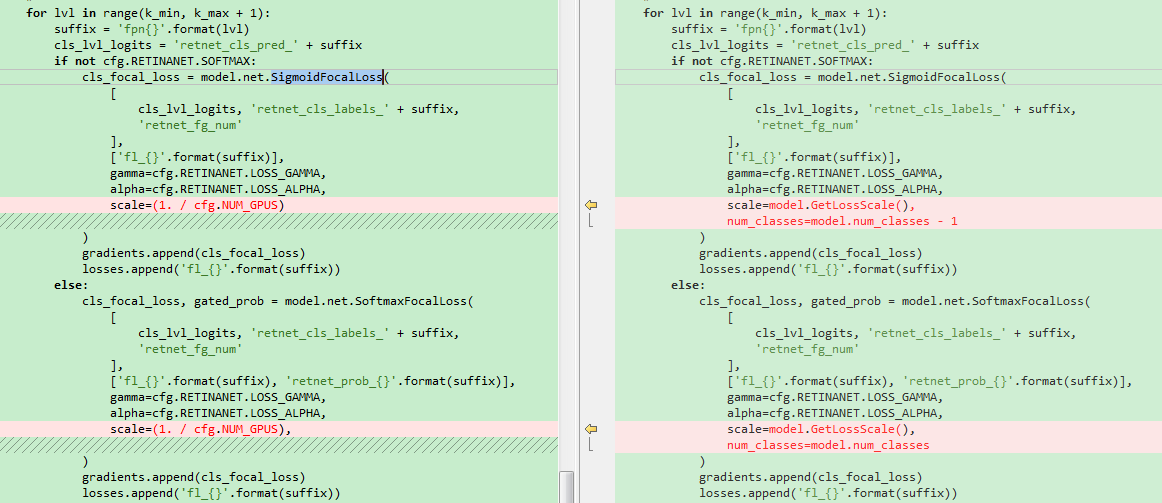


因为在caffe2 / modules / detectron / sigmoid\_focal\_loss\_op.cu中会使用num\_classes，并且根据doc默认为80，如果不传入这个参数，可能会导致奇怪的现象。这是一个阻止RetinaNet使用num\_classes！= 81进行学习的bug。

<https://github.com/facebookresearch/Detectron/commit/1624f810a4e0a5bac924c4e6abe2b31f66a986aa>

上面是作者修正这个问题提交代码的链接。在2月23号已经解决了这个问题。

因此重新更新代码，下图是新代码和老代码在[lib/modeling/retinanet\_heads.py](https://github.com/facebookresearch/Detectron/blob/021685d42f7e8ac097e2bcf79fecb645f211378e/lib/modeling/retinanet_heads.py#L280-L304)文件的差别（左边是老代码；右边是新代码）：



再次重新做实验。

## 使用新版本的detectron

### retinaNet-R-101-FPN(P2 : P7)，768尺寸

网络结构：retinaNet-101

预训练模型为R-101.pkl

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为128；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 5.0, 0.5, 0.2),6个ratios

图片大小：配置为768，896

2个GPU，batchsize总共为4。

下面是训练和测试的结果：（retinanet\_R-101-FPN\_P7\_2\_R6.yaml）

测试时的配置仍然为768，896

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 1万次 | 1.69🡪0.06 | 0.8799 | 0.9014 | 0.6747 |
| 4万次 | 1.69🡪0.04 | 0.9926 | 0.9968 | 0.7504 |
| 6万次 | 1.69🡪0.01 | 0.9956 | 0.9976 | 0.7442 |

注意：这是用新版训练，旧版测试的结果。新版测试有些问题。

测试时的配置的尺寸为512，640。这样的测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 4万次 | 1.69🡪0.04 | 0.8697 | 0.8655 | 0.6684 |
| 6万次 | 1.69🡪0.01 | 0.8699 | 0.8957 | 0.6680 |

时间测试：retinaNet-R-101-FPN

|  |  |  |
| --- | --- | --- |
| 输入大小 | 验证集(850) 单位：毫秒 | 测试集(349) 单位：毫秒 |
| 512 | 全部时间：131  前向传播时间：67 | 全部时间：124  前向传播时间：68 |
| 768 | 全部时间：208  前向传播时间：122 | 全部时间：196  前向传播时间：120 |

### retinaNet-R-50-FPN(P2 : P7)，768尺寸

网络结构：retinaNet-50

预训练模型为R-50.pkl

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为128；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 5.0, 0.5, 0.2),6个ratios

图片大小：配置为768，896

1个GPU，batchsize为2。

下面是训练和测试的结果：（retinanet\_R-50-FPN\_P7\_2\_R6.yaml）

测试时的配置仍然为768，896

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 3万次 | 1.68🡪0.1 |  | 0.9667 | 0.6884 |
| 5万次 | 1.69🡪0.03 | 0.9764 | 0.9940 | 0.7155 |
| 9万次 | 1.69🡪0.02 | 0.9879 | 0.9953 | 0.7148 |
| 28万次 | 1.69🡪0.02 | 0.9890 | 0.9954 | 0.7133 |

注意：这是用新版训练，旧版测试的结果。新版测试有些问题。

### 3．retinaNet-R-101-FPN(P2 : P7)，512尺寸

网络结构：retinaNet-101

预训练模型为R-101.pkl

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为128；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 5.0, 0.5, 0.2) ,6个ratios

图片大小：配置为512，640

2个GPU，batchsize总共为4。

下面是训练和测试的结果：（retinanet\_R-101-FPN\_P7\_2\_R6\_512.yaml）

测试时的配置仍然为512,640

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 1万次 | 1.63🡪0.1 | 0.9156 | 0.9312 | 0.6999 |
| 3万次 | 1.63🡪0.03 | 0.9848 | 0.9948 | 0.7340 |
| 6万次 | 1.63🡪0.02 | 0.9870 | 0.9956 | 0.7275 |
| 12万次 | 1.63🡪0.02 | 0.9870 | 0.9956 | 0.7274 |

注意：这是用新版训练，旧版测试的结果。新版测试有些问题。

前向传播时间：68ms；总共检测时间：130ms

### retinaNet-X-101-FPN(P2 : P7)，512尺寸

网络结构：retinanet\_X-101-64x4d

预训练模型为X-101-64x4d.pkl

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 2，COARSEST\_STRIDE为128；

RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 5.0, 0.5, 0.2), 6个ratios

图片大小：配置为512，640

2个GPU，batchsize总共为4。

下面是训练和测试的结果：（retinanet\_X-101-FPN\_P7\_2\_R6\_512.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 1万次 | 1.69🡪0.1 | 0.8882 | 0.8956 | 0.6707 |
| 3万次 | 1.69🡪0.04 | 0.9845 | 0.9917 | 0.7376 |
| 4万次 | 1.69🡪0.02 | 0.9888 | 0.9950 | 0.7435 |
| 5万次 | 1.69🡪0.01 | 0.9902 | 0.9965 | 0.7461 |

注意：这是用新版训练，旧版测试的结果。新版测试有些问题。

测试时的配置的尺寸为768，896。这样的测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 4万次 | 1.69🡪0.04 | 0.9024 | 0.9289 | 0.7141 |
| 5万次 | 1.69🡪0.01 | 0.9020 | 0.9307 | 0.7125 |

时间测试：retinaNet-X-101-FPN

|  |  |  |
| --- | --- | --- |
| 输入图大小 | 前向传播时间 | 全部预测时间 |
| 512 | 130ms | 200ms |
| 768 | 250ms | 330ms |

### 总结

下面是几个模型的时间和精度的比较：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 输入图大小 | 网络 | 前向传播时间 | 全部预测时间 | 评测指标 mAP  （训练/验证/测试） | | |
| 512 | retinaNet-X-101-FPN | 130ms | 200ms | 0.9902 | 0.9965 | 0.7461 |
| retinaNet-R-101-FPN | 68ms | 130ms | 0.9848 | 0.9948 | 0.7340 |
| SSD | 83ms | 164ms | 0.903 | 0.743 | 0.686 |
| 768 | retinaNet-X-101-FPN | 250ms | 330ms | 暂无 | 暂无 | 暂无 |
| retinaNet-R-101-FPN | 120ms | 200ms | 0.992 | 0.9968 | 0.7504 |
| retinaNet-R-50-FPN | 100ms | 170ms | 0.9879 | 0.9953 | 0.7148 |
| 416 | Yolo v3 | 23ms |  | 0.9234 | 0.7554 | 0.6759 |

下图是yolov3的测试结果：（测试网络和训练网络大小为416）

|  |  |  |  |
| --- | --- | --- | --- |
| 迭代次数 | 训练集mAP | 验证集mAP | 测试集mAP |
| 800次 | 0.0878 | 0.07838 | 0.0516 |
| 1万次 | 0.9106 | 0.7532 | 0.6508 |
| 3万次 | 0.9234 | 0.7554 | 0.6759 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 输入图大小 | 网络 | 前向传播时间 | 全部预测时间 | 评测指标 mAP  （训练/验证/测试） | | | 备注 |
| 512 | retinaNet-X-101-FPN | 130ms | 200ms | 0.9902 | 0.9965 | 0.7461 | 512图片训练  512图片测试 |
| retinaNet-R-101-FPN | 68ms | 130ms | 0.8697 | 0.8655 | 0.6684 | 768图片训练  512图片测试 |
| SSD | 83ms | 164ms | 0.903 | 0.743 | 0.686 | 512图片训练  512图片测试 |
| 768 | retinaNet-X-101-FPN | 250ms | 330ms | 0.9020 | 0.9307 | 0.7125 | 512图片训练  768图片测试 |
| retinaNet-R-101-FPN | 120ms | 200ms | 0.992 | 0.9968 | 0.7504 | 768图片训练  768图片测试 |

备注里标红的部分是训练图片的尺寸和测试图片的尺寸不一致的情况，这里可以略做参考。若是训练图片的尺寸和测试图片的尺寸一致，最终的指标应该会更高一些。