# RetinaNet训练结果记录

# ———调整FPN层，anchors，预训练模型

下面的实验都使用新版本的detectron

### 一．预训练模型为R-101.pkl

### 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）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

### 时间效果总结

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-R-101-FPN | 120ms | 200ms | 0.992 | 0.9968 | 0.7504 |
| retinaNet-R-50-FPN | 100ms | 170ms | 0.9879 | 0.9953 | 0.7148 |

### 预训练模型为基于coco训练的model\_final.pkl

### retinaNet-R-101-FPN(P2 : P7)，R6

网络结构：retinaNet-101， 预训练模型为/opt/yushan/Detectron/model\_final.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\_ft.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 2万次 | 1.69🡪0.06 | 0.9511 | 0.9664 | 0.7522 |
| 4万次 | 1.69🡪0.02 | 0.9911 | 0.9969 | 0.7587 |
| 6万次 | 1.69🡪0.01 | 0.9937 | 0.9959 | 0.7540 |
| 预训练模型：R-101.pkl | 1.69🡪0.04 | 0.9926 | 0.9968 | 0.7504 |

因为这里新训练模型的FPN层是P2~P7，预训练的model\_final.pkl对应的FPN层是P3~P7，因此预训练模型的FPN的每一个分类和回归层对应4个（由RETINANET.NUM\_CONVS设置）卷积层和最后的预测卷积层的参数并没有加载进去。

为了进一步验证预训练模型中FPN分类和回归层的参数的作用，再次使用P3~P7的FPN层，训练新模型。

### retinaNet-R-101-FPN(P3 : P7)，R6

网络结构：retinaNet-101， 预训练模型为/opt/yushan/Detectron/model\_final.pkl

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 3，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。

这样的配置会将所有的权重加载进去，包括了FPN的每一个分类和回归层前面的4个卷积层。仅仅不加载最后的一个预测的卷积层参数。

下面是训练和测试的结果：（retinanet\_R-101-FPN\_P7\_3\_R6\_ft.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 2万次 | 1.72🡪 0.03 | 0.9893 | 0.9924 | 0.7732 |
| 4万次 | 1.72🡪0.01 | 0.9942 | 0.9963 | 0.7716 |
| 6万次 | 1.72🡪0.08 | 0.9943 | 0.9966 | 0.7668 |
| 8万次 | 1.72🡪0.08 | 0.9943 | 0.9966 | 0.7667 |

### retinaNet-R-101-FPN(P3 : P7)，R3

网络结构：retinaNet-101， 预训练模型为/opt/yushan/Detectron/model\_final.pkl

FPN中 MULTILEVEL\_RPN为 True，并且RPN\_MAX\_LEVEL为 7，RPN\_MIN\_LEVEL为 3，COARSEST\_STRIDE为128；RETINANET中 SCALES\_PER\_OCTAVE为 3， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 0.5), 3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\_3\_R3\_ft.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 2万次 | 1.41🡪0.06 | 0.9775 | 0.9866 | 0.7715 |
| 4万次 | 1.41🡪0.02 | 0.9827 | 0.9878 | 0.7707 |
| 6万次 | 1.41🡪0.01 | 0.9835 | 0.9883 | 0.7660 |

### retinaNet-R-101-FPN(P2 : P7)，R6

因为之前验证过，P2层和增加anchors都会带来性能的提升。

预训练模型依旧为/opt/yushan/Detectron/model\_final.pkl，，网络结构为retinaNet-101。FPN为P2~P7，ASPECT\_RATIOS为 (1.0, 2.0, 3.0, 5.0, 0.5, 0.2), 6个ratios。RETINANET中 SCALES\_PER\_OCTAVE为 4， ANCHOR\_SCALE为 4；

图片大小：配置为768，896，2个GPU，batchsize总共为4。

这次修改代码，加载所有预训练模型的权重，将预训练模型的FPN的每一个分类和回归层对应4个（由RETINANET.NUM\_CONVS设置）卷积层的参数加载进去。（没有P2层结构的权重和最后预测的卷积层权重）。

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

 这次结果保存在retinanet\_R-101-FPN\_P7\_2\_R6\_ft\_my的目录下。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 2万次 | 1.61🡪0.03 | 0.9884 | 0.9967 | 0.7679 |
| 4万次 | 1.61🡪0.01 | 0.9965 | 0.9979 | 0.7570 |
| 6万次 | 1.61🡪0.01 | 0.9965 | 0.9980 | 0.7549 |

### 预训练模型为R-101.pkl

### retinaNet-R-101-FPN(P3 : P7)，R3

预训练模型为R-101.pkl，模型其他参数和上面的一样。FPN为P3~P7，COARSEST\_STRIDE为128；SCALES\_PER\_OCTAVE为 3， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 0.5), 3个ratios；图片大小：配置为768，896，2个GPU，batchsize总共为4。

下面是预训练模型为R-101.pkl，训练和测试的结果：（retinanet\_R-101-FPN\_P7\_3\_R3.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 2万次 | 1.61🡪0.06 | 0.9590 | 0.9760 | 0.7117 |
| 4万次 | 1.61🡪0.02 | 0.9778 | 0.9843 | 0.7204 |
| 6万次 | 1.61🡪0.01 | 0.9803 | 0.9855 | 0.7222 |

### retinaNet-R-101-FPN(P2 : P7)，R3

预训练模型为R-101.pkl，模型其他参数和上面的一样。FPN为P2~P7，COARSEST\_STRIDE为128；SCALES\_PER\_OCTAVE为 3， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 0.5), 3个ratios；图片大小：配置为768，896，2个GPU，batchsize总共为4。

下面是预训练模型为R-101.pkl，训练和测试的结果：（retinanet\_R-101-FPN\_P7\_2\_R3.yaml）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | Loss | 训练集 (3406) | 验证集 (850) | 测试集 (349) |
| 2万次 | 1.63🡪0.01 | 0.9525 | 0.9748 | 0.7252 |
| 4万次 | 1.63🡪0.01 | 0.9775 | 0.9855 | 0.7424 |
| 6万次 | 1.63🡪0.01 | 0.9800 | 0.9863 | 0.7401 |

CALES\_PER\_OCTAVE为 3， ANCHOR\_SCALE为 4，ASPECT\_RATIOS为 (1.0, 2.0, 0.5), 3个ratios；图片大小：配置为768，896，2个GPU，batchsize总共为4。

### 四．总结

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验 | 实验细节  coco的模型：P3~P7, 3个ratios | Loss | 训练集  (3406) | 验证集  (850) | 测试集  (349) |
| 1 | 预训练模型：coco训练的模型  P3~P7, 3个ratios | 1.41🡪0.02 | 0.9827 | 0.9878 | 0.7707 |
| 2 | 预训练模型：coco训练的模型  P3~P7, 6个ratios | 1.72🡪0.01 | 0.9942 | 0.9963 | 0.7716 |
| 3 | 预训练模型：coco训练的模型  P2~P7, 6个ratios  不加载FPN预测层的所有卷积层 | 1.69🡪0.02 | 0.9911 | 0.9969 | 0.7587 |
| 4 | 预训练模型：coco训练的模型  P2~P7, 6个ratios  加载FPN预测层的所有卷积层 | 1.61🡪0.03 | 0.9884 | 0.9967 | 0.7679 |
| 5 | 预训练模型：R-101.pkl  P3~P7, 3个ratios | 1.61🡪0.02 | 0.9778 | 0.9843 | 0.7204 |
| 6 | 预训练模型：R-101.pkl  P2~P7, 3个ratios | 1.63🡪0.01 | 0.9775 | 0.9855 | 0.7424 |
| 7 | 预训练模型：R-101.pkl  P2~P7, 6个ratios | 1.69🡪0.04 | 0.9926 | 0.9968 | 0.7504 |

实验5和实验6对比，当只有ResNet的权重加载时，不修改anchors，最终增加了P2层的指标会高一些，主要表现在测试集上的提升。

实验6和实验7对比， 当只有ResNet的权重加载时，最终根据数据集分布修改的anchors，的指标会更高一些。

实验5，实验6，实验7对比，当只有ResNet的权重加载时，最终根据数据集分布修改的anchors，并增加P2层的指标（实验7）最高的。

即：增加P2层和修改anchors都会带来性能的提升，在测试集上，增加P2层比修改anchors带来的性能提升更大一些。

实验4中P2层没有预训练权重，FPN的分类回归层（P2~P7）的权重是加载了预训练模型的分类回归层(P3~P7)的权重。

实验3和实验4对比，coco预训练的模型所有能加载的权重都加载（实验4）要比不加载FPN的分类回归层权重（实验3）的指标高一些。

实验4和实验1，实验2对比，可以看出，如果P2层和后面的分类回归层不能提供好的预训练模型的权重，反而不如使用和作者预训练模型一样网络结构的效果。

实验1和实验2对比，当coco预训练的模型的所有权重都加载时，修改anchors，最终根据数据集分布修改的anchors指标会稍微高一些。

实验1和实验5对比，即使用coco预训练的模型和使用R-101.pkl预训练模型对比，最终使用coco预训练的模型的效果更好（coco训练的模型的所有权重都加载）。

实验3和实验7对比，即使用coco预训练的模型和使用R-101.pkl预训练模型对比，最终的结果相差不大。（coco训练的模型并没有加载FPN的每一个预测大层对应的所有卷积层权重）。

**最优效果的实验：新训练的模型结构和和作者提供的coco模型结构一样，仅根据数据修改anchors，而且权重全部加载（实验2）。**

**【总结】**

当coco预训练模型的所有权重都加载时，使用coco预训练模型的效果要比使用R-101.pkl预训练模型的效果更好。

当coco预训练模型的FPN层的权重不加载时，使用coco预训练模型的效果和使用R-101.pkl预训练模型的效果基本一样。

如果新模型的P2层和后面的分类回归层不能提供好的预训练模型的权重，最好使用和作者预训练模型结构一样的网络。

不管是使用coco预训练模型还是使用R-101.pkl预训练模型，根据新训练的数据集分布修改的anchors，都会在训练集和验证集上提升效果。