1.数据载入

数据载入部分的代码主要见./data/dataset.py中的类Dataset与TestDataset。

数据载入部分的逻辑如下：

从VOC数据集中获得img, bbox, label --data/voc\_dataset.py/VOCBboxDataset.get\_example

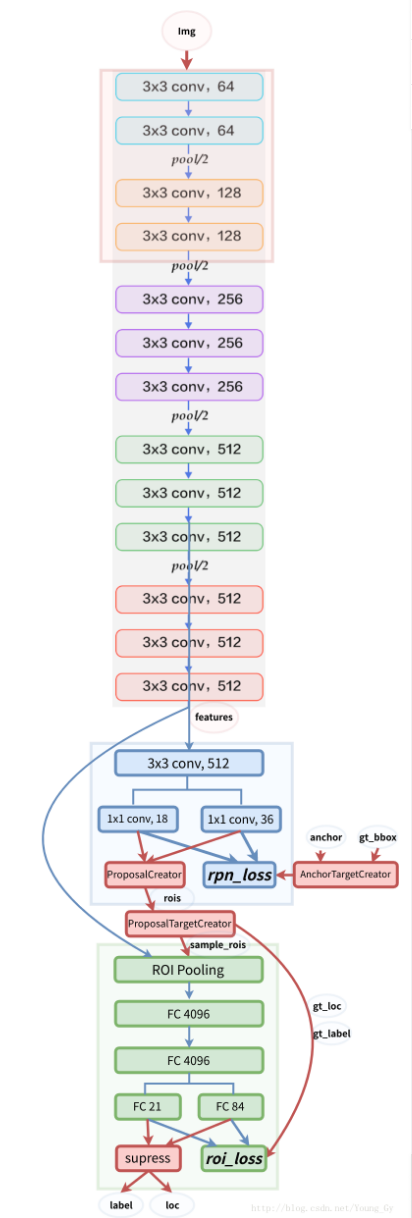
将img, bbox进行放缩（放缩的目的是让图片处于合适的大小，这样预先指定锚框才有意义）--data/dataset.py/preprocess

将img进行标准化正则处理 --data/dataset.py/preprocess

如果是训练阶段，将img翻转以增加训练数据 --data/dataset.py/Transform

2.网络结构

FasterRCNN的网络结构如下图所示：



FasterRCNN结构的代码主要见./model.faster\_rcnn.py，其结构包含三大部分：

预训练的CNN模型 decom\_vgg16 --model/vgg16.py

rpn网络RegionProposalNetwork --model/rpn.py

roi及以上网络VGG16RoIHead --model/roi\_module.py

下面，将以放缩后大小为[1, 3, 600, 800]的图片为例针对每个部分分别介绍。图像类别共计21类（包含背景）

2.1 预训练的CNN模型

该部分代码见./model/vgg16.py。

输入：图片，大小[1, 3, 600, 800]   
输出：特征图features，大小[1, 512, 37, 50]

其逻辑如下：

载入预先训练好的CNN模型VGG16。

将模型拆分为两部分extractor, classifier。其中，extractor的参数固定。

图片通过extractor可以得到特征图features。根据extractor中池化参数可知图像通过extractor缩小了16倍。

--model/faster\_rcnn.py

2.2 rpn网络

该部分代码见./model/rpn.py。

输入：特征图features，大小[1, 512, 37, 50]

输出：

rpn\_locs：rpn对位置的修正，大小[1, 16650, 4]

rpn\_scores ：rpn判断区域前景背景，大小[1, 16650, 2]

rois：rpn筛选出的roi的位置，大小[300， 4]

roi\_indices：rpn筛选出的roi对应的图片索引，大小[300]

anchor：原图像的锚点，大小[16650, 4]

其中，16650是放缩后的图像所产生的所有锚点（37\*50\*9），每个锚点都对应了一个rp。通过 rpn\_scores以及nms可以得到筛选后的大小为300的roi。

其逻辑如下：

对特征图features以基准长度为16、选择合适的ratios和scales取基准锚点anchor\_base。（选择长度为16的原因是图片大小为600\*800左右，基准长度16对应的原图区域是256\*256，考虑放缩后的大小有128\*128，512\*512比较合适） --utils/anchors.py/generate\_anchor\_base

根据anchor\_base在原图上获得anchors。--utils/anchors.py/get\_anchors

对特征图features采用卷积得到rpn\_locs和rpn\_scores

根据anchors和rpn\_locs获得修正后的rp --utils/anchors.py/get\_rois\_from\_loc\_anchors

对rp进一步修正获得rois和roi\_indices，修正包括超出边界的部分截断、移除太小的、nms。

NMS --utlis/py\_nms.py

2.3 roi及以上网络

该部分代码见./model/roi\_module.py。

输入：

features：特征图，大小[1, 512, 37, 50]

rois：rpn筛选出的roi的位置，大小[300， 4]

roi\_indices：rpn筛选出的roi对应的图片索引，大小[300]

输出：

roi\_cls\_locs：roi位置的修正，大小[300， 84]

roi\_scores：roi各类的分数，大小[300， 21]

其逻辑如下：

通过RoIPooling2D将大小不同的roi变成大小一致，得到pooling后的特征，大小为[300, 512, 7, 7] --model/roi\_module.py/ROI\_forward\_backward

接入预训练的CNN模型引入的classifier

分别接入全连接得到roi\_cls\_locs、roi\_scores

3.训练

训练部分的代码主要见./trainer/trainer.py中的FasterRCNNTrainer中的train\_step函数。

训练部分的核心是loss如何求取。

loss求取前网络的步骤如下：

预训练CNN特征提取：输入img到extractor获得features

rpn网络得到roi：输入features到rpn获得rpn\_locs, rpn\_scores, rois, roi\_indices, anchor

抽样anchor:输入anchor和bbox到AnchorTargetCreator获得gt\_rpn\_loc, gt\_rpn\_label。该步骤的含义是得到正负比例和位置合适的anchor，用于训练Faster\_rcnn\_rpn部分。

--model/utils/rpn\_gt\_loc\_label.py --n\_sample=256, pos\_iou\_thres=0.7, neg\_iou\_thres=0.3, pos\_ratio=0.5

pos\_label = iou > pos\_iou\_thres

neg\_label = iou < neg\_iou\_thres

抽样roi：输入rois，bbox，label到ProposalTargetCreator获得sample\_roi, gt\_roi\_loc, gt\_roi\_label。该步骤的含义是得到正负例比例和位置合适的roi，用于训练Faster\_rcnn head部分。

--model/utils/roi\_sample.py n\_sample=128, pos\_ratio=0.25, pos\_iou\_thresh=0.5, neg\_iou\_thresh\_lo=0, neg\_iou\_thres\_hi=0.5

pos\_label = iou > pos\_iou\_thresh

neg\_label = neg\_iou\_thres\_hi > iou > neg\_iou\_thresh\_lo

head网络得到roi的位置修正与分数：输入features,sample\_roi,sample\_roi\_index得到roi\_cls\_loc, roi\_score

各个loss求取的方式如下：

rpn\_loc\_loss：已知rpn\_loc，需要先根据anchor和bbox得到真实的gt\_rpn\_loc和gt\_rpn\_label。该处loss的计算只考虑前景，所以根据rpn\_loc,gt\_rpn\_loc,gt\_rpn\_label计算L1-LOSS即可。

rpn\_cls\_loss：根据rpn\_score和gt\_rpn\_label计算二分类的交叉熵即可。

roi\_loc\_loss：已知roi\_loc，在sample roi的过程中已获得gt\_roi\_loc, gt\_roi\_label。根据roi\_loc,gt\_roi\_loc,gt\_roi\_label计算L1-LOSS即可。

roi\_cls\_loss：根据roi\_score和gt\_roi\_label计算多分类的交叉熵即可。

整体的loss为以上各loss相加求和。

4.测试

训练部分的代码主要见./model/faster\_rcnn.py中的FasterRCNNTrainer中的predict函数。

Nms作用在最终的检测结果，参数如下所示。

nms\_thresh=0.3

score\_thresh=0.7

object\_score > score\_thresh and object\_thresh > 0.7

其步骤如下：

图片预处理 --data/dataset.py/TestDataset

预训练CNN特征提取：输入img到extractor获得features

rpn网络得到roi：输入features到rpn获得rpn\_locs, rpn\_scores, rois, roi\_indices, anchor

head网络得到roi的位置修正与分数：输入features,rois,roi\_indices得到roi\_cls\_loc, roi\_score

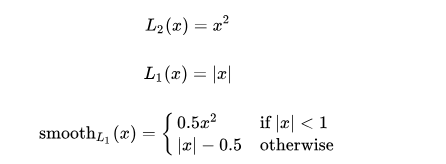
得到图片预测的bbox：输入roi\_cls\_loc、roi\_score、rois，采用nms等方法得到预测的bbox。

1. 名词解释

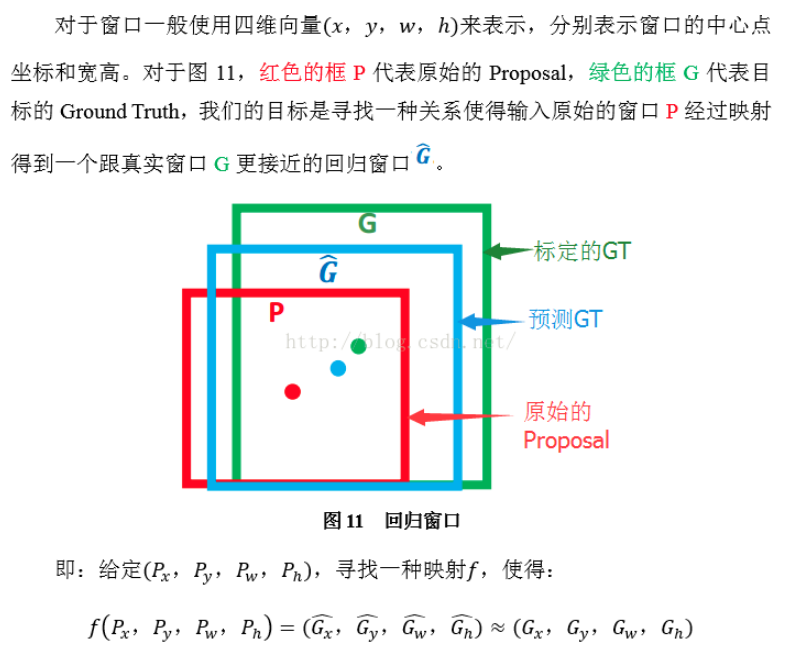
nms原理



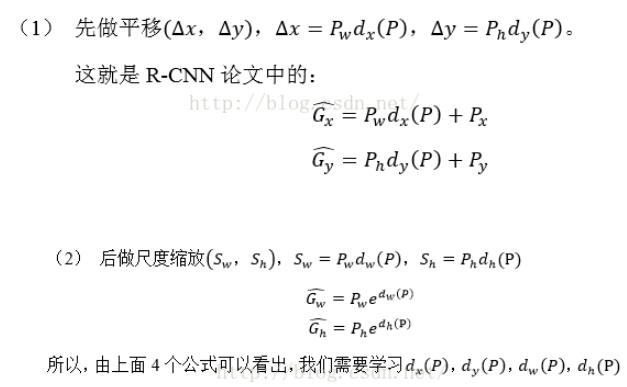
l1\_smooth\_loss原理



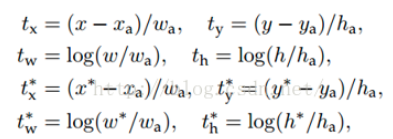
loc\_bbox and bbox\_loc原理



**Loc\_bbox:**获取bounding box



#### **bbox\_loc:** bounding box回归



训练过程：

