Faster rcnn caffe 代码详解

符号说明：GT：ground truth

1.模型的流程：

- 单张图片进来先 resize
- resize 过后的图片进入基础网络得到最后一层的特征层：
  conv5([1,h,w,256])
- conv5 进入 RPN，通过 1*1 的卷积得到 anchors 的分类和回归的预测
  分类的输出 shape：[1,h,w,2*9]，2 表示前景和背景，9 表示 anchor 数目
  回归的输出 shape：[1,h,w,4*9],4 为坐标，9 为 anchor 数目
- 在 rpn.anchor_target_layer 层中，制作 RPN 层的标签。
  （训练时才有）
  1. 生成 anchors([K,A,4],K=w*h,A=9) → reshape 到[K*A,4]
     → 去除在图片之外的 anchors ，这时剩余的 anchor 为[N,4]
  2. 为 anchor 打分类标签(shape 为[N,1])，规则为：
     a.与 GT 的 IOU 最大的几个 anchor 标为 1(前景)。
     b.与 GT 的 IOU 最大值在 0.7~1 之间的标为 1。
     c.与 GT 的 IOU 最大值在 0~0.3 的标记为 0(背景)。
     d.没有被标记的 anchor 标记为-1(不关心)。
     e.如果前景数目超过 128,则将多余的 anchor 标记为-1
     f.如果背景数目超过 256-num(前景)，则将多余的标为-1
  3. 为 anchor 打回归标签（shape 为[N,4]）：
     计算在图像内的 anchor（[N,4]）到与之 IOU 最大的 GT 的偏差。
  4. 计算 inside_weight(shape 为[N,4])：
     只有前景的 anchor 才会计算 anchor 回归的 loss，所以这个 weight 就是为了标记前景样本。
  5. 计算 outside_weight(shape 为[N,4]):
     outside_weight 是前景和背景的一个加权。
  6. 将打好的标签和 weight 映射到[K*A,~]空间。
  7. 把[K*A,~]reshape 到对应的输出 shape。
- 计算 RPN 层的 loss
- 生成 proposal
  1. 将之前的 rpn_bbox_pred（shape 为[1,h,w,4*9]）reshape 到
     [h*w*9,4]→ 用其调整 anchors 得到的结果称为 proposal
     (shape 为[h*w*9,4])

2. 将 proposal 裁剪到图像大小之内
3. 根据之前的 rpn_cls_score(shape 为[1,h,w,2*9])，取前景得分 score（shape 为[1,h,w,9]）→ reshape 到[h*w*9,1]。
4. 去除尺寸（长，宽）小于阈值的 proposals，剩下的 shape 为[N1,4], 和其前景得分的 shape[N1,1]。
5. 根据 proposal 的前景得分排序，选取出前 6000 个 proposal，这时的 proposal.shape=[6000,4],score.shape=[6000,1]
6. 对剩下的 6000 个框进行 NMS
7. 剩下的框，根据得分选取前 2000 个框作为 RPN 层的输出。输出 2000 个框的位置信息，即输出的 shape 为[2000,4]

- rpn.proposal_target_layer 层。根据 RPN 层的产生的 2000 个候选框选出 128 个 ROI。制作其分类和回归的标签，还有两个回归的权重（inside_weight,outside_weight）
  1. 计算 proposals 与 GT 的 IOU，IOU 大于 0.5 的为前景，小于 0.5 大于 0.1 的为背景。前景不超过 32 个，背景不超过 128-#前景。
  2. 根据选出的前景背景的 index，为其打上分类标签，背景打为 0,前景为与其最大 IOU 的 GT 的标签。labels.shape = [128,1]
  3. 根据选出的前景背景的 index,索引出其坐标，再与其最接近的 GT 算出 bbox_target，bbox_target.shape=[128,4]。
  4. 将 bbox_target 做成网络预测的 shape，并生成 inside_weight。输入为 labels 和 bbox_target。输出的 shape[128,4*num_class], inside_weight 的 shape 也为[128,4*num_class]，是前景的一个 mask。
  5. outside_weight 与 inside_weight 相同。
- ROIPooling: 用上一层产生的 rois，和基础网络出来的 conv5 特征，计算 rois 的特征。即使 roi 的大小会不一样，但是经过这层之后的特征都是一样的了
- 两个全连接层
- 全连接之后，接一个全连接产生分类预测，接另外一个全连接层产生回归预测。
- 用之前做的 roi_bbox_target 和 roi_labels,inside_weight, outside_weight,产生分类和回归的 loss