1.数据预处理



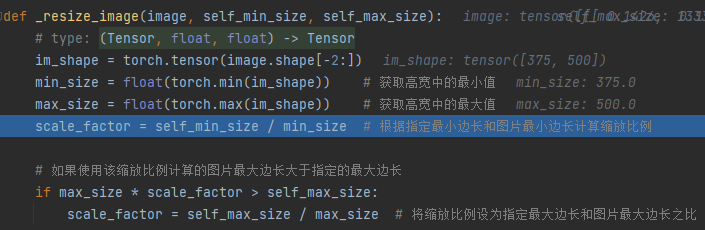
1.1对图像进行标准化

1.2.对图像进行resize(resize之后宽高中的最小值为800)：

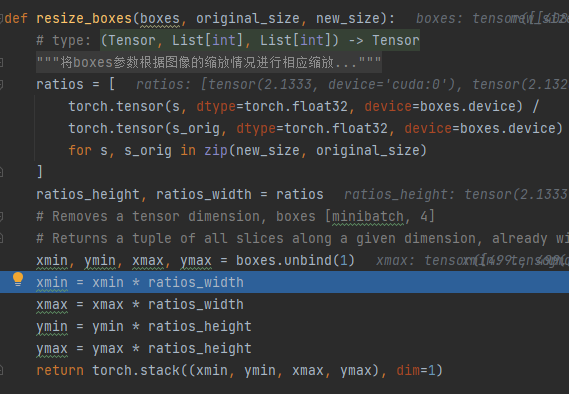
获取图像高宽中的最小值和最大值

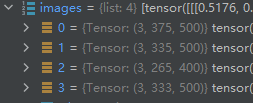
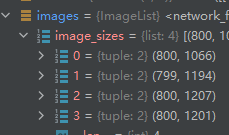
缩放比例 = 800 / 宽高中的最小值

resize后宽高中的最大值 = 图像宽高的最大值 \* 缩放比例



1.3.根据图像的缩放比例修改bbox的坐标

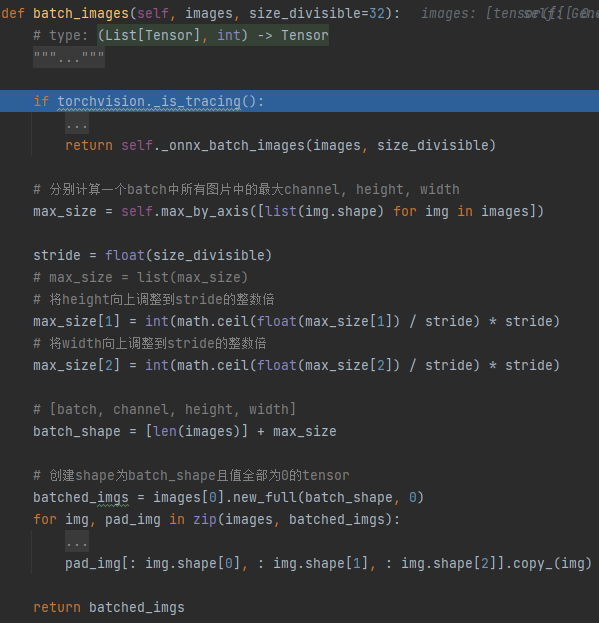


图片原尺寸

resize之后的尺寸

1.4.resize之后的图片打包成一个batch输出：将每张图的宽高向上取整到32的整数倍。

.

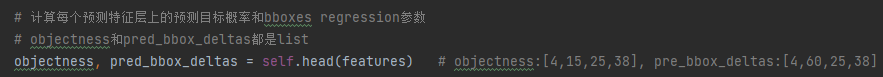
2.将数据预处理后的一个batch输入backbone

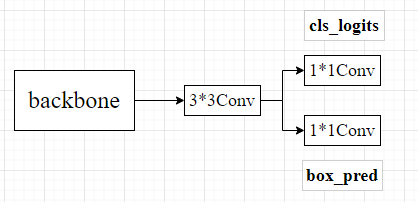


3.将经过backbone得到的特征输入rpn网络



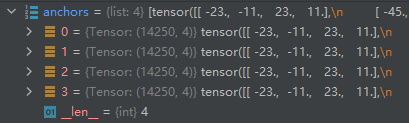
3.1经过一个3\*3卷积层分别输入到两个1\*1的卷积中，这两个1\*1卷积分别预测目标得分(前景或是背景)和bbox回归参数，第二个维度15对应anchor的数量



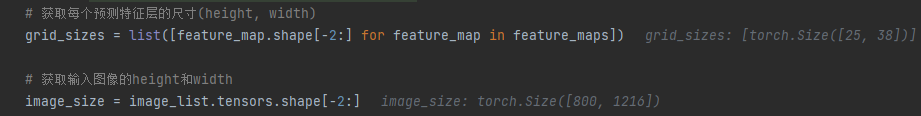


3.2.生成一个batch图像的所有的anchor信息

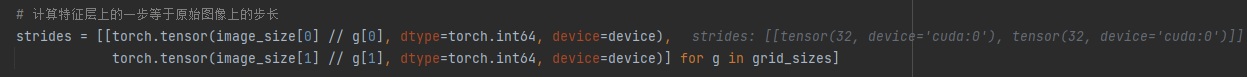




获取输入图像的尺寸(向上取整到32倍数的大小)和特征层的尺寸(经过backbone得到的特征向量的大小)



计算图像和特征层之间的比例获得在图像上和在特征层上步长之间的比例=32



根据提供的anchor的size和aspect\_ratios生成anchor模板





假设图片的面积为S，高宽比为ratios

=> =>

生成3种缩放比例\*5种anchor尺寸后的anchor共15种



生成的anchor都是以(0,0)为中心的



15个anchor的中心坐标：

tensor([[ -22.6274, -11.3137, 22.6274, 11.3137],

[ -45.2548, -22.6274, 45.2548, 22.6274],

[ -90.5097, -45.2548, 90.5097, 45.2548],

[-181.0193, -90.5097, 181.0193, 90.5097],

[-362.0387, -181.0193, 362.0387, 181.0193],

[ -16.0000, -16.0000, 16.0000, 16.0000],

[ -32.0000, -32.0000, 32.0000, 32.0000],

[ -64.0000, -64.0000, 64.0000, 64.0000],

[-128.0000, -128.0000, 128.0000, 128.0000],

[-256.0000, -256.0000, 256.0000, 256.0000],

[ -11.3137, -22.6274, 11.3137, 22.6274],

[ -22.6274, -45.2548, 22.6274, 45.2548],

[ -45.2548, -90.5097, 45.2548, 90.5097],

[ -90.5097, -181.0193, 90.5097, 181.0193],

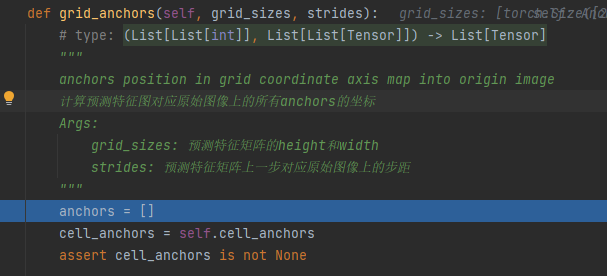
[-181.0193, -362.0387, 181.0193, 362.0387]], device='cuda:0')

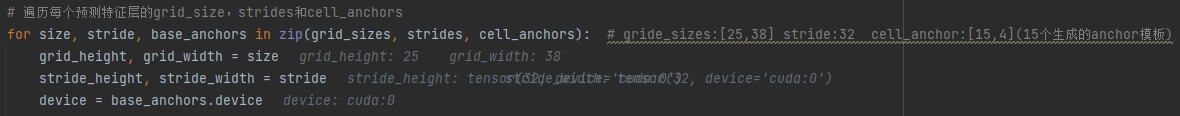
生成5\*3共15个anchor模板

计算所有anchor映射到原图上的坐标信息

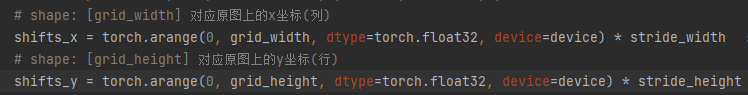


计算预测特征图对应原始图像上的anchor的坐标





计算特征图对应原图上的x坐标和y坐标



shift\_x:

tensor([ 0., 32., 64., 96., 128., 160., 192., 224., 256., 288., 320., 352., 384., 416., 448., 480., 512., 544., 576., 608., 640., 672., 704., 736., 768., 800., 832., 864., 896., 928., 960., 992., 1024., 1056., 1088., 1120., 1152., 1184.], device='cuda:0')

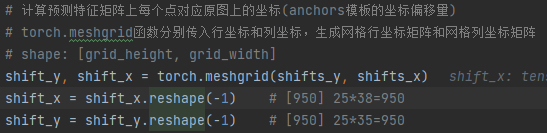
shift\_y:

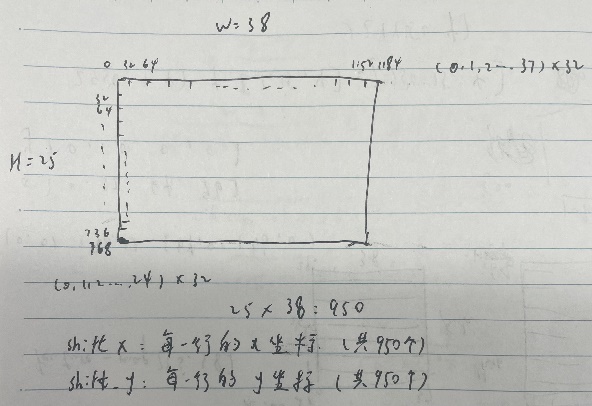
tensor([ 0., 32., 64., 96., 128., 160., 192., 224., 256., 288., 320., 352.,

384., 416., 448., 480., 512., 544., 576., 608., 640., 672., 704., 736.,

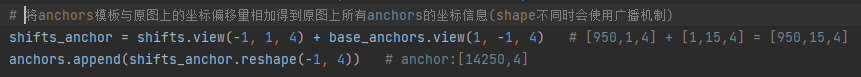
768.], device='cuda:0')

计算预测特征图上每个点对应原图上的坐标

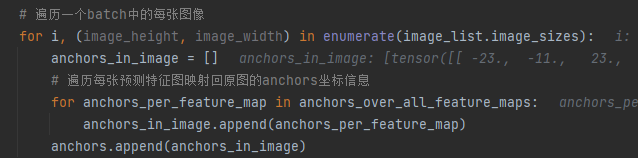
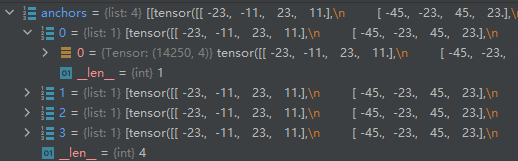




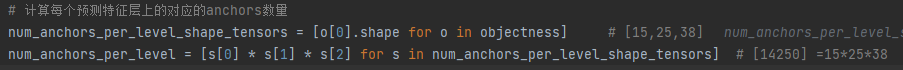
将anchor模板与原图的坐标偏移量相加得到原图上所有anchor的坐标信息



遍历一个batch中的每个图像，batch=4，得到4组anchor



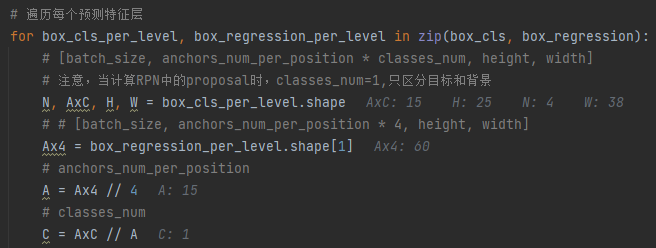
3.3计算每个特征图上anchor的数量



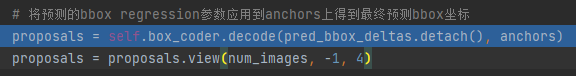
3.4 调整内部tensor格式以及shape

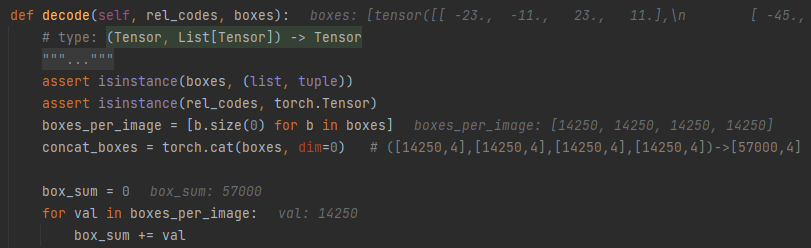
objectness：[4,15,25,38]->[57000,1] pred\_bbox\_deltas：[4,60,25,38]->[57000,4]



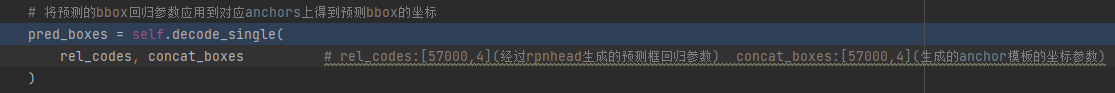


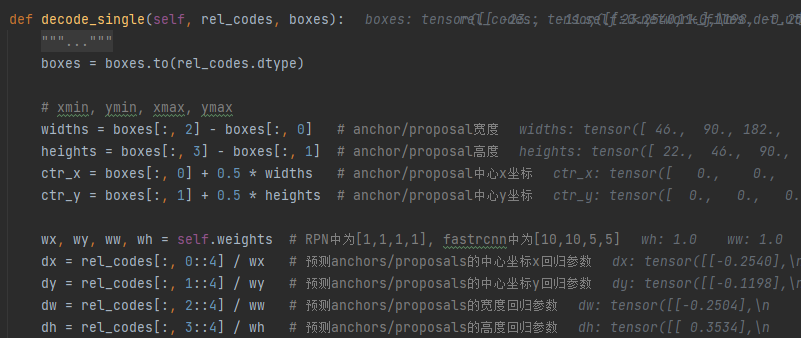
3.5将预测的bbox regression参数应用到anchors上得到最终预测bbox坐标

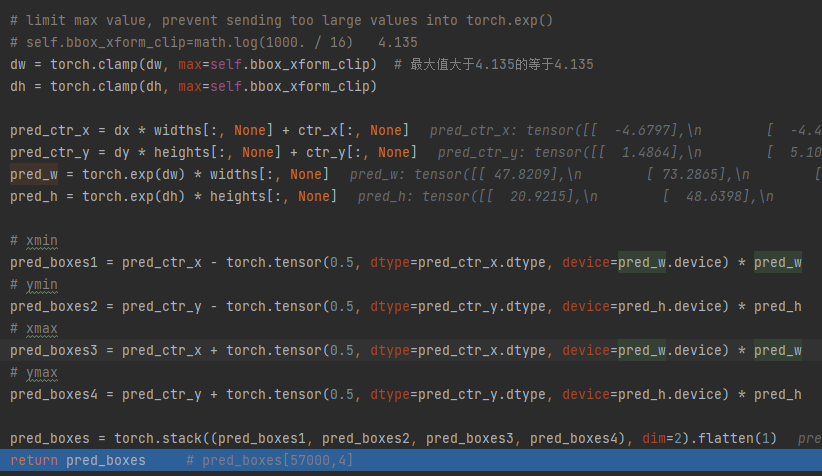




将预测的bbox回归参数应用到对应anchors上得到预测bbox的坐标







xmin:pred\_boxes1 = pred\_ctr\_x -0.5 \* pred\_w

ymin:pred\_boxes2 = pred\_ctr\_y -0.5 \* pred\_h

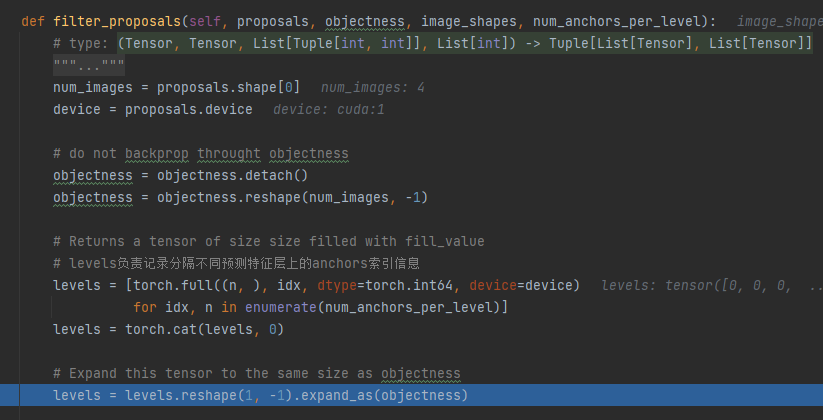
xmax:pred\_boxes3 = pred\_ctr\_x +0.5 \* pred\_w

ymax:pred\_boxes1 = pred\_ctr\_x +0.5 \* pred\_h

得到预测框的左上和右下坐标

3.6筛选小的boxes框，nms处理，根据预测概率获取前post\_nms\_top\_n个目标

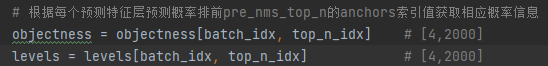




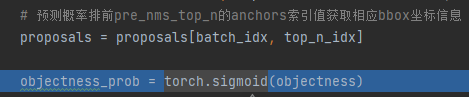
获取每张预测特征图上预测概率排前pre\_nms\_top\_n的anchors索引值，取2000个



根据每张特征图上预测概率前pre\_nms\_top\_n的索引值获取对应的概率值



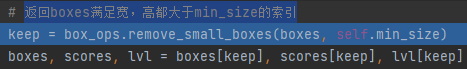
预测概率排前pre\_nms\_top\_n的anchors索引值获取相应bbox坐标信息



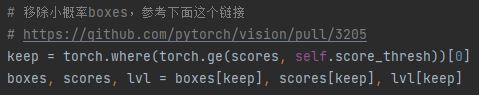
调整预测的boxes的信息，将越界的坐标调整到图片上



返回boxes满足宽，高都大于min\_size的索引



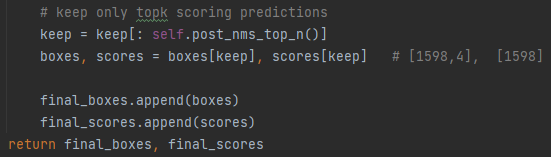
移除小概率boxes



非极大值抑制(NMS)



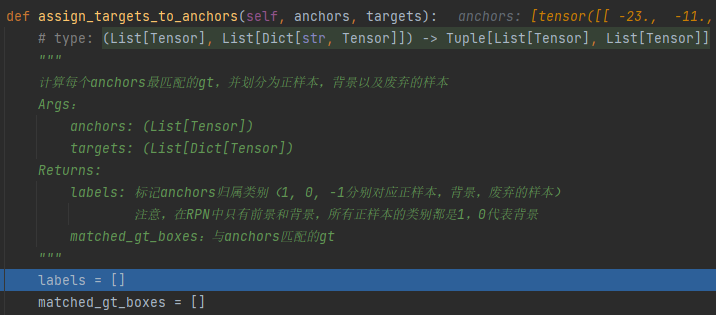
得到最终框的坐标的得分



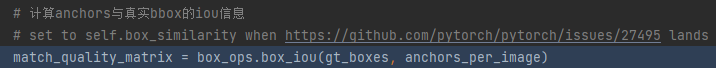
3.7计算每个anchors最匹配的gt，并将anchors进行分类，前景，背景以及废弃的anchors

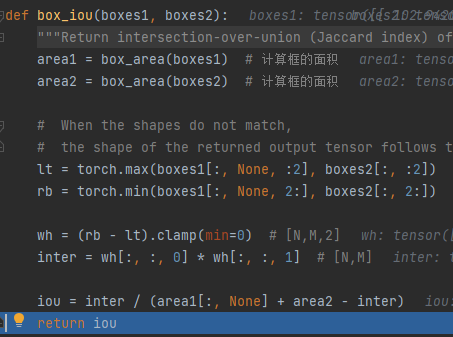


计算每个anchor最匹配的GT,并划分为正样本，背景和负样本



计算anchor与真实bbox的iou值





1.通过调用函数 box\_area(boxes1) 和 box\_area(boxes2) 计算两组矩形框的面积，其中 box\_area() 函数用于计算矩形框的面积。

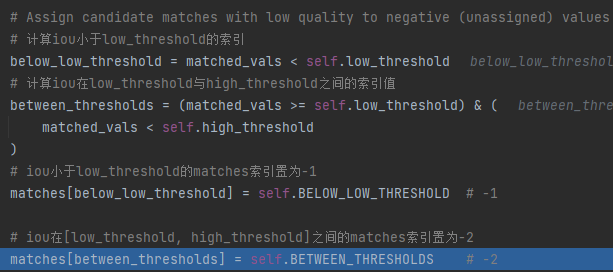
2.使用广播的方式，将两组矩形框的坐标按照左上角和右下角的方式进行比较，从而获得每对矩形框的左上角和右下角坐标的最大值和最小值。这个步骤用于计算两个矩形框的重叠部分的左上角和右下角坐标。

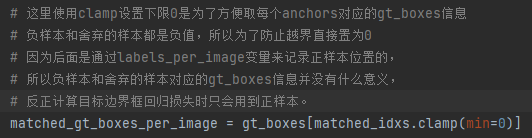
3.通过计算重叠部分的宽度和高度，得到每对矩形框的重叠部分的面积。

4.最后，通过公式 iou = inter / (area1[:, None] + area2 - inter) 计算每对矩形框的 IoU 值。其中 inter 是重叠部分的面积，area1 和 area2 分别是两组矩形框的面积。

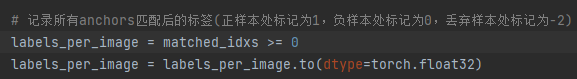
计算每个anchor与gt匹配iou最大的索引

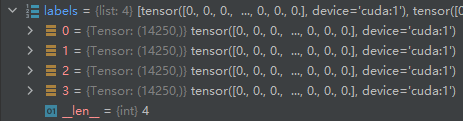
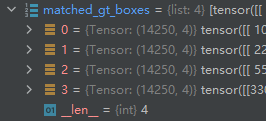






记录所有anchors匹配后的标签

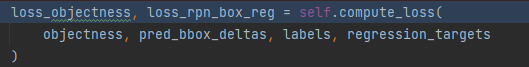


3.8结合anchors以及对应的gt，计算regression参数



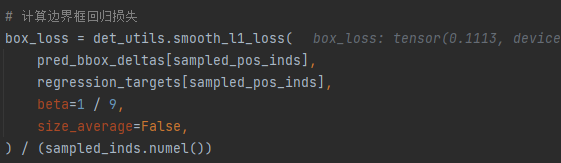
3.9计算类别损失(前景或背景)和边界框回归参数损失



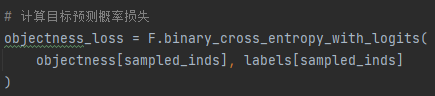
按照给定的batch\_size\_per\_image, positive\_fraction选择正负样本



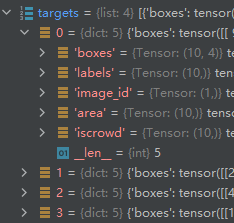
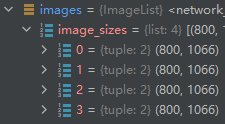
计算边界框回归损失

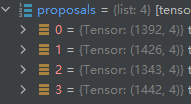


计算目标预测概率损失



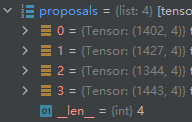
4. 将rpn生成的数据以及标注target信息传入fast rcnn后半部分





4.1划分正负样本，统计对应gt的标签以及边界框回归信息

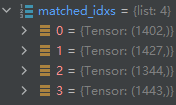
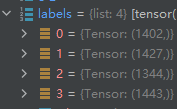


 将gt\_boxes拼接到proposal后面



为每个proposal匹配对应的gt\_box，并划分到正负样本中





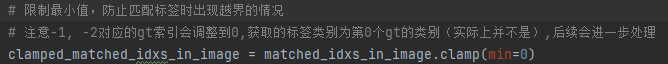
计算proposal与每个gt\_box的iou重合度



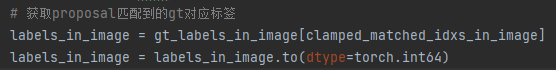
计算proposal与每个gt\_box匹配的iou最大值，并记录索引，iou < low\_threshold索引值为 -1， low\_threshold <= iou < high\_threshold索引值为 -2(介于正样本和负样本之间的重复样本)



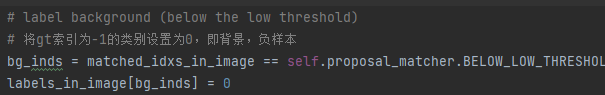
限制最小值，防止匹配的标签出现越界的情况，-1，-2对应的gt索引会调整到0，暂时当作第0个gt的类别，后续会处理



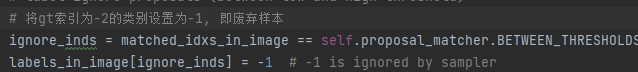
获取proposal匹配到的gt对应标签



将gt索引为-1的类别设置为0，即背景，负样本

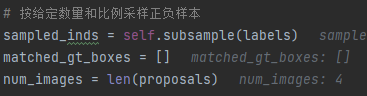
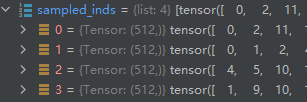


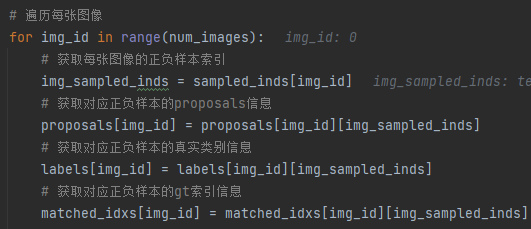
将gt索引为-2的类别设置为-1，即废弃样本



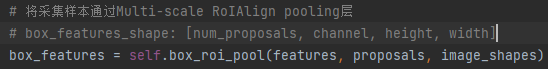
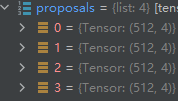
##############################

按给定数量和比例采样正负样本

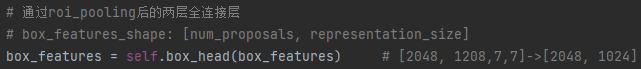


4.2将采集出来的样本送入Multi-scale RoIAlign pooling层





4.3 将得到的box\_feature送入roi\_pooling的后两个连接层.

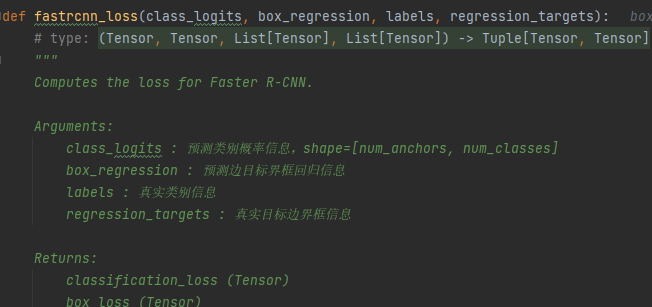


4.4 然后预测目标类别和边界框回归参数



4.5计算faster\_rcnn\_loss





5.对网络的预测结果进行后处理(将bbox还原到原图像尺度上)

