训练fasterRCNN时，必须先确保有VOC格式的数据集

首先，将fasterrcnn网络框架构建好，并加载预训练权重减少不必要的训练时间

训练需要分成两部分：

1. 先训练RPN，即训练边框初步回归分支与判断是否包含物体分支，使网络具有提取建议框的能力；注意到，在这一阶段训练，每个位置点的9个先验框是人为事先分配好的；
2. 利用提取出的建议框，与共享特征层求交，训练网络最后的分类分支与边框精细回归分支

关于正负样本要注意，伴随着训练，正负样本分别出现在3个不同的阶段内：

第一阶段是RPN的先验框是否包含物体（包含具体物体的先验框为正样本，包含背景为负样本），分别随机选择128个正负样本训练，可以达到初步预测出建议框的目的

第二阶段是判断建议框与真实框的重合度，（可以线性回归的是正样本，必须非线性回归的是负样本），可以线性回归的意思是上一阶段选出的建议框与真实框IOU比较大，则认为可以线性回归到真实框，同样分别随机选择128个正负样本训练，这样训练可以使网络获得更少的建议框，也方便用线性回归方式进行准确调整

第三阶段是最终选出的建议框中，所含物体的类别，多个二分类构成的多分类，这里的正负样本就是分解出的一个个二分类样本

下面先看一下smoothL1函数：

首先回归问题中，损失函数一般为：

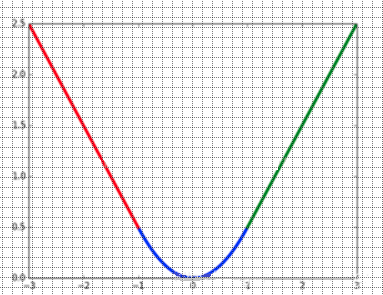
这会导致当计算的误差较大时惩罚变得过高，如果直接使用线性的会导致在0点函数不连续，求导会出错，影响收敛

所以使用smoothL1函数进行回归：



一般

可视化后为：



现在开始训练流程说明：

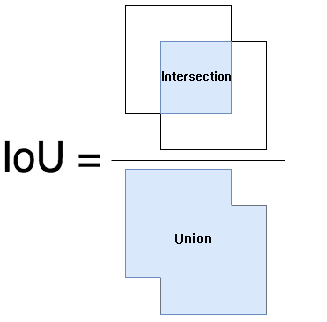
太多的图片导致内存不能容纳，所以借助数据生成器提供源源不断的批量数据，数据生成器读取图片的时候，会对图片进行随机的光照补强等图像增强操作，目的是为了使数据集多样化，提高网络的泛化能力

读取数据集之后，数据生成器会返回两类数据：

增强后的图片，真实边框的坐标信息

在共享特征层上获得所有先验框，每个先验框与每个真实框计算一次iou，iou大于0.7的设置为正样本，小于0.3的设置为负样本，在两个值中间的框要忽略，不加入训练

插入iou：将两个框按照如下方式计算，即为iou



将正负样本分批传给RPN训练，损失函数为RPN的分类损失与回归损失之和，分类只有两类：包含物体类，背景类；优化器为Adam，学习率为1e-5

分类损失使用交叉熵损失函数：



其中，，0代表负样本，1代表正样本，，为正负样本的总数，为了平衡正负样本数量，会从所有正负样本中，随机选择128个正样本，128个负样本，因此，就来自上面所说传入RPN的正负样本，经过RPN自己的预测得到正负样本的分类概率，将预测与真实标签计算交叉熵损失便可以通过优化算法降低损失

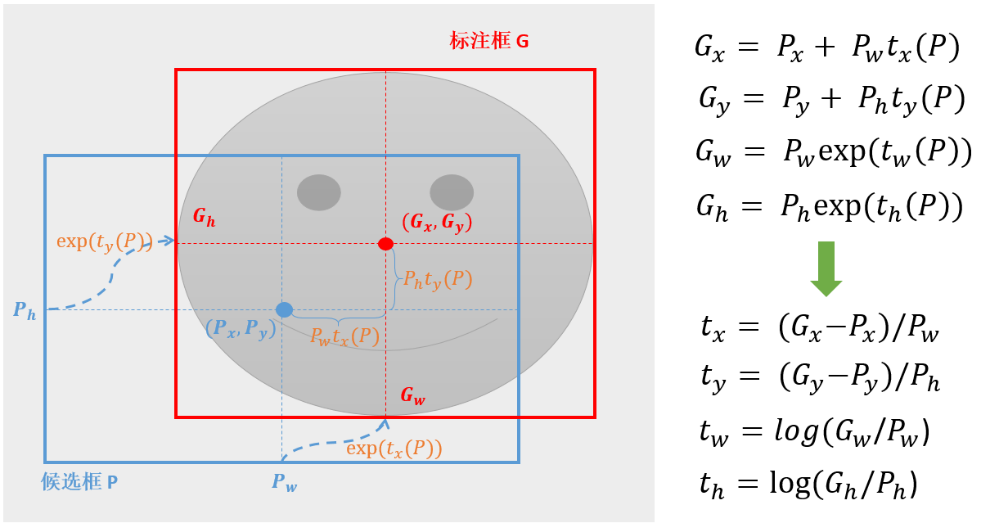
RPN的回归损失为：



，代表RPN预测的偏移量；

代表实际参与训练的先验框与真实框的真实偏移量；

偏移的计算方式如下，P代表预测框的位置，G代表真实框的位置：



注意到smoothL1与前面的系数相乘后即表示只有正样本才具有非零的损失值，如果训练的128个先验框包含背景，即负样本，损失将不考虑，因为调整背景框是没有意义的

是回归训练中，正负样本数之和，一般正负样本总共会选出256个，一般为1，如果当正负样本数设置比较大，比如上千，的值就会变得很小，此时设为5或更大的数，就可以突出回归损失值，使回归损失数量级和分类损失数量级一致，兼顾两者，才能使训练有效

在这个epoch内的一次训练中，RPN完成了一代训练，现在要使用RPN对生成器提供的新一批数据预测出粗略的建议框，将这些建议框再次与真实框计算求出iou，当iou大于0.6时，认为两个框比较相似，用线性回归到真实框是可以实现的，如果iou小于0.6，必须按照非线性回归才可以调整到真实框，但这已经不是我们要计算的任务了，所以大于0.6的为正样本，小于0.6的为负样本，此时又得到了一次正负样本，将正负样本传入训练，现在RPN具有了提取理论上可以支持线性回归的建议框

因此在这里，又出现了一次分类损失函数，和回归损失函数，函数的形式与上面的一样，只是计算的对象变成：

分类的目标是可线性回归的框和必须非线性回归的框，回归的计算只考虑可以线性回归的框

这个epoch中的一次训练还没有结束，现在使用RPN再次预测一批数据，获得了更加合理的建议框，将建议框映射到共享特征层大小上，截取特征层的每一个区域，现在就把这些截取出来的特征层们看做“一个一个的原图片经过特征提取网络（resnet或者VGG）获得的特征层”，我们要做的就是预测它们分别对应的标签，分类使用交叉熵损失；除此之外，在训练分类的同时，训练4个位置回归的参数，目的是对建议框进行更加细致的调整

在最后分类层的损失函数，从之前简单的二分类交叉熵损失变成多分类的交叉熵损失，最后输出类别使用的是softmax层，softmax层可以将输出变成概率分布的形式：

比如在某一个框即某一个目标预测时，网络分类层的分类分支输出为

经过softmax函数变换，得到对应的概率为：



现在，把class\_num个对应的概率组成一个概率分布，记为，举个例子，假设真实的标签为，这是由标签生成的One-hot编码，为1的位置确定了类别是什么，即类别的索引，记这个One-hot编码为概率分布

因此，交叉熵的计算为：



假如只有三个类别要分类，正确标签的编码为，softmax输出概率分布，

损失计算为：



边框的回归还有一次，这里选出的已经是合理的建议框，所以回归损失为：



是建议框的个数

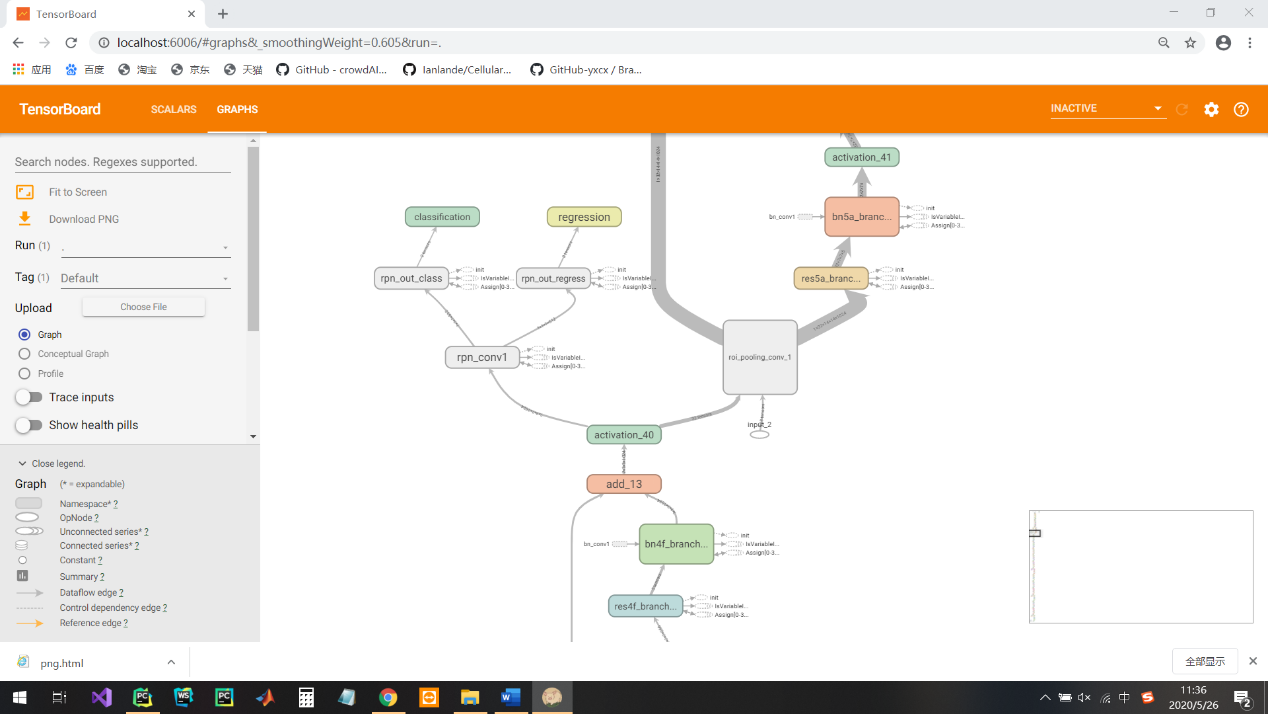
前面说过网络的训练分两个阶段，第一阶段训练RPN，第二阶段才训练最后的分类层（分类层包括分类分支，回归分支）

因此训练RPN的综合损失函数为提取出合理的建议框到roipooling层之前的损失函数之和：分成4个损失，2个分类损失，2个回归损失；

提取出合理的建议框后，就开始训练分类层，后面的损失函数之和就是分类层的综合损失，只有2个损失，1个分类损失，1个回归损失

在很多个epoch后，损失减小，训练结束，权重保存到.h5文件

使用labelimg标注口罩数据及后得到训练结果，使用tensorboard查看计算图：

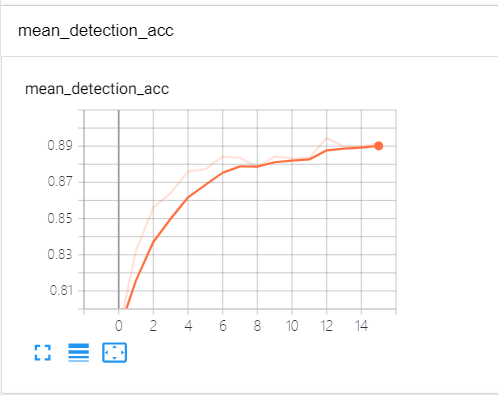


fasterRCNN整体比较庞大，只展示出部分

损失函数之和的减小过程为：



平均的检测准确率为：



使用权重进行测试：

