**《专业方向前沿课》线上实验/课程大作业 报告**

**(2022-2023 学年第 2 学期)**

**第二单元 目标检测**

**学生姓名：**

**提交日期： 2023 年 7 月 22 日 学生签名：**

|  |  |  |  |
| --- | --- | --- | --- |
| **学 号** |  | **任课教师** |  |
| **学 院** |  | **专业班级** |  |
| **课程名称** |  |  |  |
| **教师评语：** | | | |
| **本论文成绩评定：** **分** | | | |

**说 明**

1、本次大作业报告需要配套完成线上实验的ipynb文件一同提交

2、ipynb需要包括实验手册1~4节所对应实验一和实验二的全部内容，包括各实验的中间结果，最后结果，以及开放题的实现与结果。

3、本课程大作业报告，重点摘录ipynb笔记中关实验一、实验二的运行情况，结果分析，以及两个实验开放性问题的回答，包括改进方案和实验结果的分析说明。

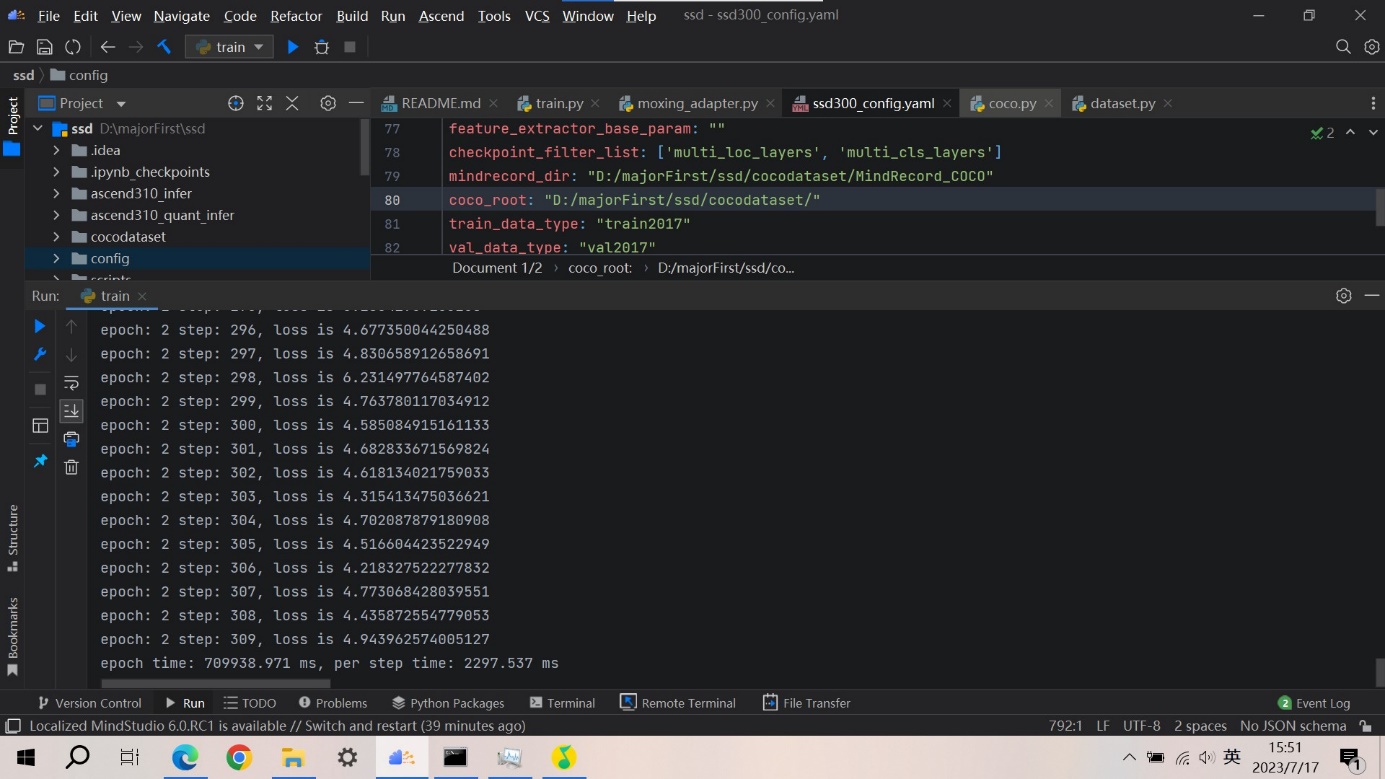
4、针对开放性问题的分析说明，本报告需要完整阐述所用模型或网络方法的修改方案，具体的参数设置，训练及测试的配置情况，数据使用情况，实验结果与分析评价。

6、请记录实验完成的费用情况。实验二如果费用不足，请参照实验一的流程，部署本地mindspore，做阶段性的训练验证。

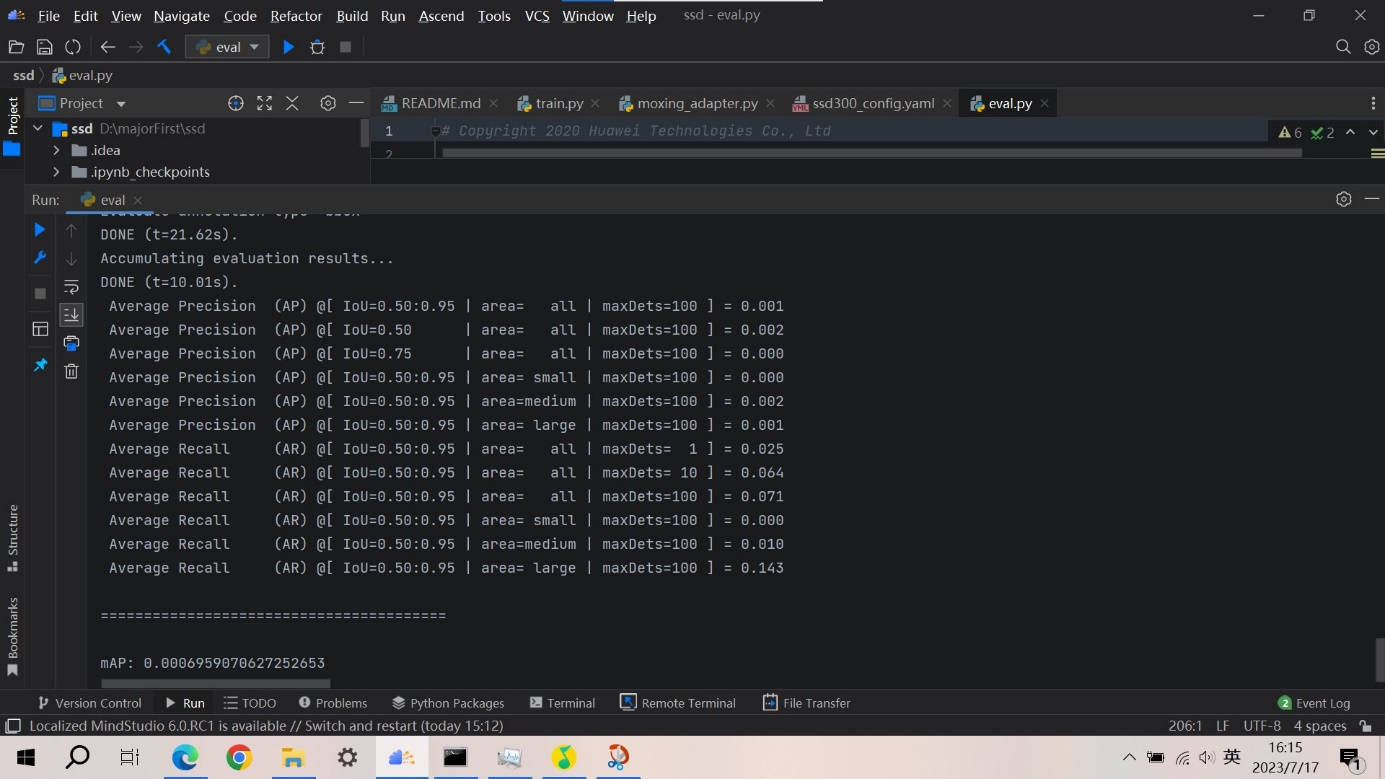
1. **实验一 复现SSD网络的目标检测**

**1、本地复现**

1.1 训练

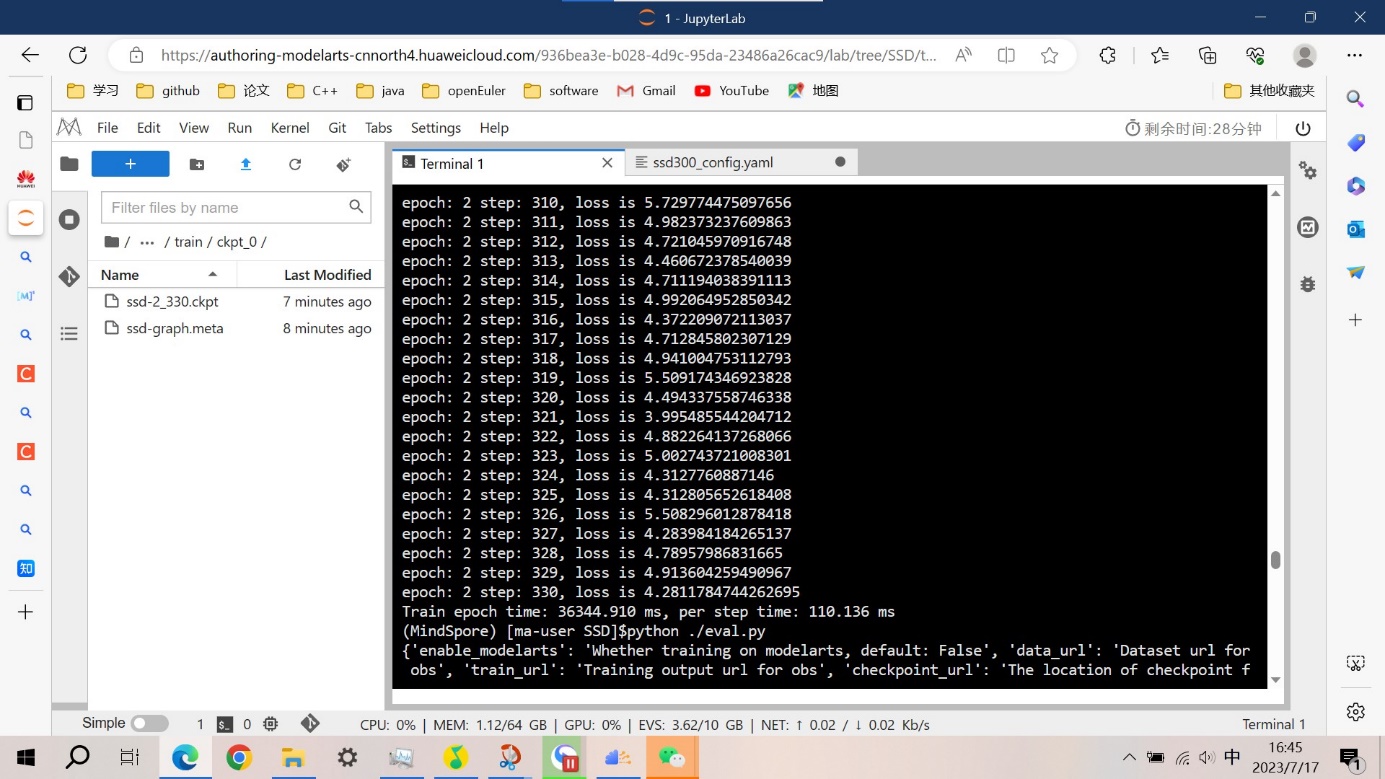


1.2 评估

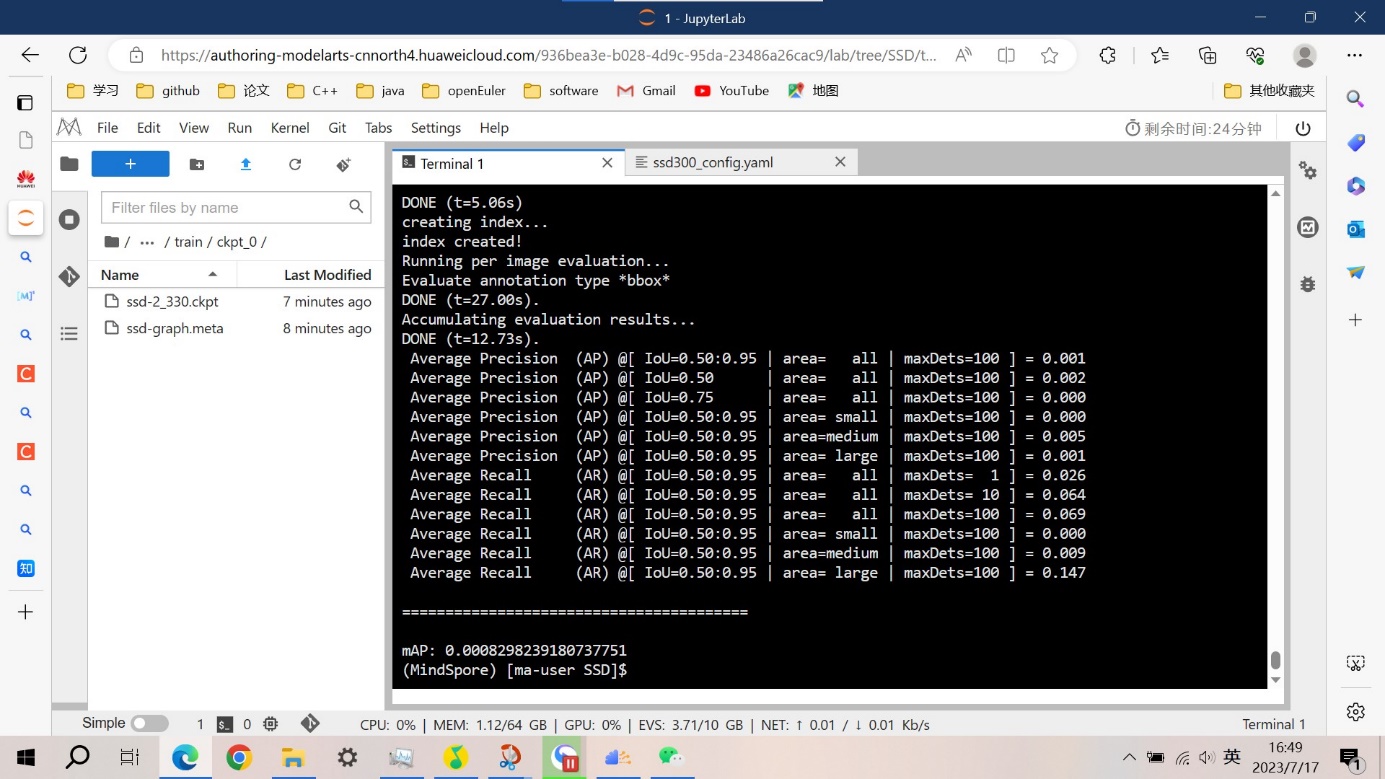


**2、云端复现**

2.1 训练



2.2 评估



1. **实验二 复现FasterRCNN网络的目标检测**

1、训练

Ipynb文件运行时控制台网页崩溃了，因此截取了日志中的输出记录。

注：batch\_size 2, epoch 12。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

2、评估

图形用户界面, 应用程序

描述已自动生成

3、实验总结：

关于eval.py没有运行出结果，我在gitee查了一下类似的issue，大致说是eval.py中的这段代码为空。

文本

低可信度描述已自动生成

Gitee上mindspore官方给的解决方案是使用多卡进行训练。附连接([【C类交付模型faster-rcnn】评估得到的results.pkl.bbox.json是空的 这样导致没有评估结果 · Issue #I3U5FB · MindSpore/mindspore - Gitee.com](https://gitee.com/mindspore/mindspore/issues/I3U5FB?_from=gitee_search))

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成 因为是云端实验，再者结合每个epoch中loss均为NaN，在查阅了一下资料，有一种可能是学习率过大，便分别修改将base\_lr从原来的0.04依次修改为0.01和0.001，batch\_size均为2，仅训练一个epoch，结果的到的loss仍然是NaN。运行出的评估得到的results.pkl.bbox.json是空的。

学习率为0.01：



日志输出：



学习率为0.001：



日志输出：

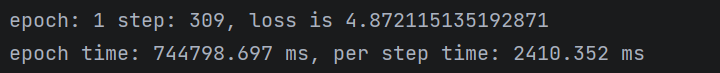


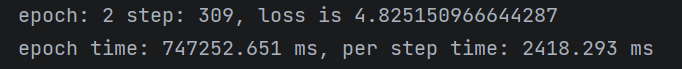
1. **实验一开放问题**

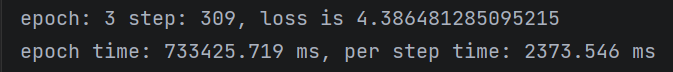
**1、尝试用CPU训练模型，运行若干个epoch，评估并与GPU训练时间进行对比。**

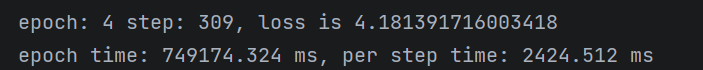
答：在复现过程中，考虑到本地算力有限，分别采用了了2个epoch和5个epoch进行训练，并且batch\_size均为19。2个epoch的训练结果如一中的复现所示。5个epoch如下图所示。可以发现，每个epoch的训练时间变化幅度很小，但明显GPU训练花费的时间比CPU小很多，节约了将近一半的训练时间。并且mAP的数值GPU也比CPU的大，说明GPU训练效果比CPU好。

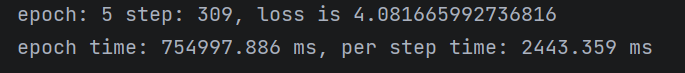
本地：五个epoch分别用时如下所示











评估：

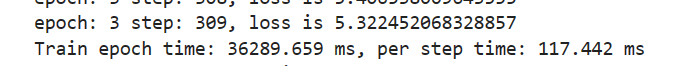
电脑萤幕的截图

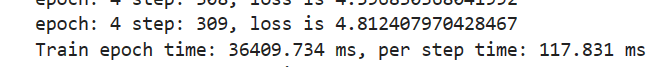
描述已自动生成

云端：五个epoch分别用时如下所示











评估：

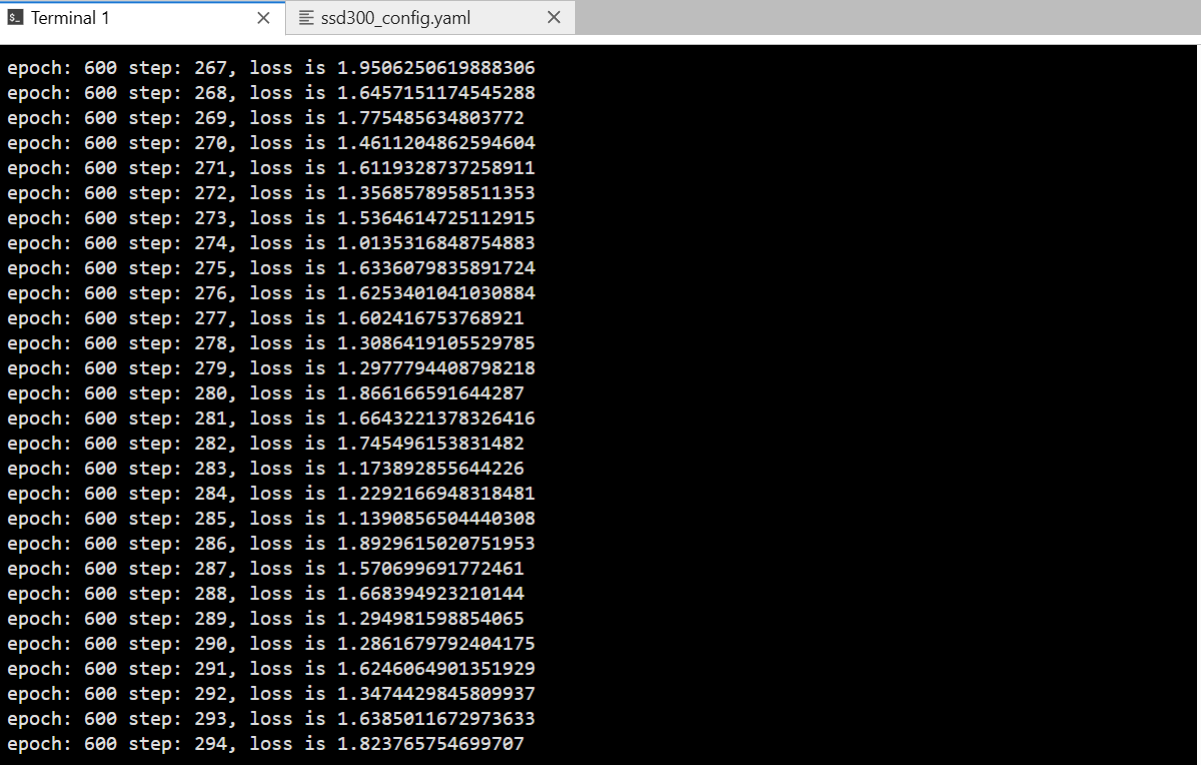
表格

中度可信度描述已自动生成

2、GPU训练模型的参数调整与评估

1）、最佳epoch次数

六百次epoch过后，每个step的loss变化已经比较少，趋于收敛。



2）、增大 batch\_size：我将batch\_size从16增加到32，并运行5个epoch，与第一问中云端运行5个epoch，batch\_size为16进行对比。以下是batch\_size为32的五个epoch输出。可以发现第一个epoch因为要生成许多初始文件，会比其他epoch耗时较多，尽管如此，batch\_size比较大的训练其花费的时间比较小的少了将近五分之三。此后每个epoch训练花费的时间，两个batch\_size花费的相近，但是batch\_size大的per step time也比batch\_size小的增加了。推测batch\_size与per step time存在一个线性关系。











1. **实验二开放题**

**1、rcnn、Fast-RCNN、Faster-RCNN是一个系列的文章，阅读这三篇文章，你觉得他们各自之间有什么关系，又有什么改进？**

答：

**RCNN**：主要通过提取多个Region Proposal(候选区域)来判断位置，作者认为以往的对每个滑动窗口进行检测算法是一种浪费资源的方式。在RCNN中，不再对所有的滑动窗口跑算法，而是只选择一些窗口，在少数窗口上运行CNN。主要有以下流程：

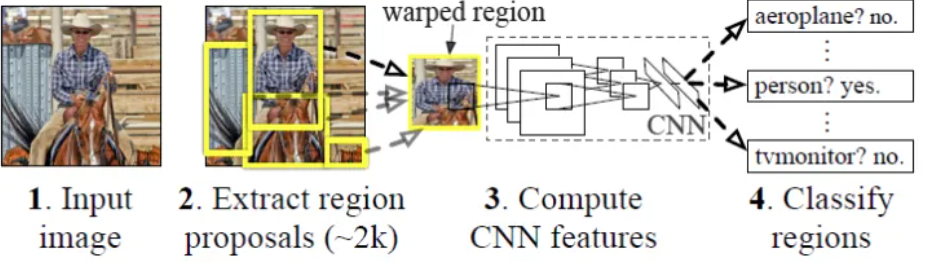
1)输入图像

2) 利用selective search对图像生成1K~2K的候选区域（region proposal），这个量比传统的算法要少得多。具体一点，选出region proposal的方法是运行图像分割算法，对于分割算法跑出来的块，把它作为可能的region proposal输出。

3) 提取特征：将region proposal resize为统一大小，送进去掉了softmax的CNN，对每个候region proposal提取特征

4) 对区域进行分类：对从CNN output出来的特征向量送进每一类的SVM分类, 如果我有十个类别，那么每个region proposal要跑10个SVM，得到类别。这里为什么要用SVM而不是softmax，有一种说法是为了解决样本不均衡的问题，另外是早期神经网络还不如现在这样发达，当时SVM还是比较领先的分类器。

5) 修正：对CNN output的特征向量（这个特征向量和第4步中拿去喂给SVM的是一个向量）做回归（左上角右下角的四个坐标），修正region proposal的位置。



**Fast R-CNN:**RCNN可以明显地感受到它的计算量是非常大的，毕竟要对每个候选区域都进行特征计算。冗余计算太多了，毕竟候选区域高度重叠。同时又不是端到端的训练，还麻烦。内存占用：需要储存多个SVM分类器和bounding box 回归器。对输入图片的大小有硬性要求。这种情况下，二代目fast R-CNN出现了，它的流程是：

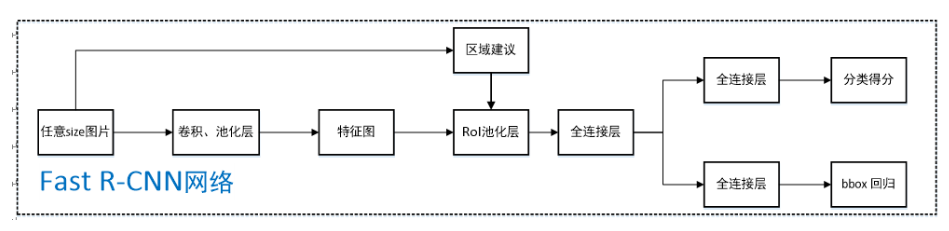
1) 将任意size的图片输入CNN，得到特征图。在RCNN中，先生成region proposals再做卷积，相当于做了多次卷积，浪费时间。

2) 对原始图片使用selective search算法得到约2k region proposals（相当于RCNN的第一步）。

3) 在特征图中找到每一个region proposals对应的特征框。在ROI池化层中将每个特征框池化到统一大小。

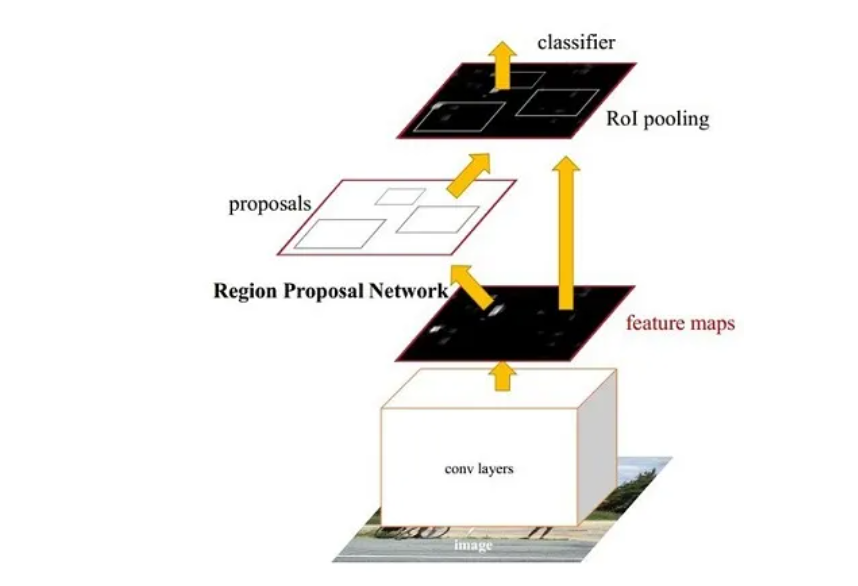
4) 统一大小的特征框经过全连接层得到固定大小的特征向量，分别进行softmax分类（使用softmax代替了RCNN里面的多个SVM分类器）和bbox回归。

Fast R-CNN组合了classification和regression, 做成single Network，实现了端到端的训练，实际上它相对RCNN最大的改进是抛弃了多个SVM分类器和bounding box回归器的做法，一起输出bbox和label, 很大程度上提升了原始RCNN的速度。



**Faster RCNN：**在Fast RCNN的基础上，Faster RCNN在性能上又有了进步。Faster RCNN将特征抽取(feature extraction)，proposal提取，bounding box regression，classification都整合在了一个网络中，使得综合性能有较大提高，在检测速度方面尤为明显。对比起它哥哥Fast-RCNN， 其实最重要的一点就是使用RPN（下面会详细解说）来代替原来使用分割算法生成候选框的方式，极大的提升了检测框生成速度。总地来说，Faster RCNN对Fast RCNN的改进点在于获得region proposals的速度要快很多。

其结构大致如图：

****

1) 提取特征：输入固定大小的图片，进过卷积层提取特征图feature maps

2) 生成region proposals: 然后经过Region Proposal Networks(RPN)生成region proposals。该层通过softmax判断anchors属于foreground或者background，再利用bounding box 回归修正anchors获得精确的proposals（候选区域）。

3) ROI Pooling: 该层的输入是feature maps和proposals，综合这些信息后提取proposal feature maps

4) Classification: 将Roi pooling生成的proposal feature maps分别传入softmax分类和bounding box regression获得检测物体类别和检测框最终的精确位置。

**2、思考在不同场景的数据集应该如何调整config.py超参数，会带来什么影响**

常见调整超参数的方法是在两个参数构成的矩形内，随机取样，理想状态下，任何两点都不同行不同列，可以保证没有浪费的试验，同时可以迅速锁定最优区域，更快地找到最佳的超参数组合。

对于不同场景的数据集，我认为可以选取少量样本先进行调试，先把少量样本训练至收敛后，在加大样本的训练。不同数据集中的数据不完全相同，需要逐步的去调参，两个数据集中的参数不一定完全相似。

1. **机时费用情况**

**补充：请在此记录本次大作业完成时的机时消费金额情况：￥ \*\*\*.\*\* 元**

**本次实验花费金额：￥245.33元**