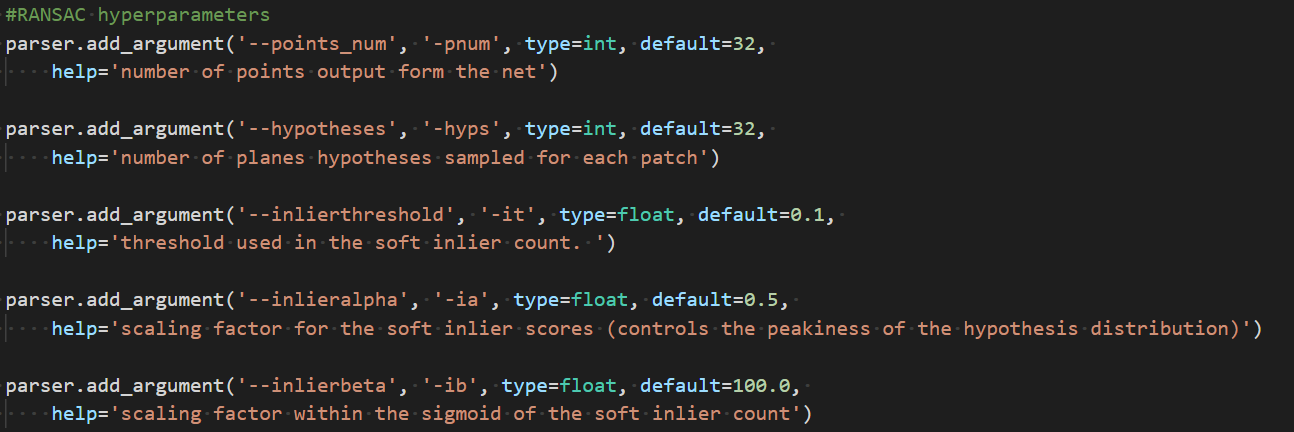
**Train.py：**基本上是从pcpnet的改过来的



这几个参数是RANSAC部分的，第一个是网络生成点的个数，第二个是选择的平面的个数，下面两个是控制评分函数的，最后一个是控制softmax分布情况的。

**DSAC.py:** ransac部分。其返回的5项分别为1.随机选择32平面平均的loss，2.最好的loss，3.预测的法向，4.网络输出的32个点，5.mask

**calculate.py:**利用训练好的模型，根据测试数据生成法向，保存到指定路径下。

**Evaluate.py:**根据calculate.py的结果进行评估。

**draw.py：**从evaluate.py那个改过来的，利用visdom进行可视化，在浏览器上显示结果。里面写的是循环一直输出，如果太多的话浏览器那边会卡，我为了省事没写太复杂，输出的patch差不多了就手动控制它停止。

**关于实验：**

Train.py中设置的是对于数据迭代了2000轮，一般来说loss会在100轮左右就能降到0.1左右，之后会继续降，但是比较缓慢，大概1500轮后基本就比较稳定了，但是我大部分时候还是让他跑完2000轮后再做评估的。

跑完2000轮还是需要一定时间的，所以在大部分情况下，我都是在训练过程中和之前结果保存同样的轮数的模型做一个简单的对比（用他们分别处理同样的patch，看看loss，可视化结果什么的），就能大致看出当前的实验究竟是有提升还是下降，如果有提升才迭代完2000轮做评估。

**之前做过的尝试：**

改变输出点的数量对于结果没什么太大影响，但是输出点的数越多所需选择平面的次数越多，速度也就会变慢。

对于特征维数尝试从1024增加到2048维，在低噪声时有提升，高噪声时下降，特征增加参数也就多了速度也会变慢。

两分支Mask尝试过乘到坐标点上，乘到特征上，拼到坐标点上，都没什么效果

单分支mask也是没什么效果

当然mask都是在之前RANSAC的基础上加的。

对于多尺度尝试过nesti-net的方式，训练的速度特别慢，结果也不是很好，个人认为这种方式不可取

如果之后mask还是没有多大提升的话，我觉得还是应该在对于每一个patch在进网络前都确定一个合适的半径。

还有如果要是尝试迭代的话，是否可以将网络输出的点再拼回输入的点上进网络迭代呢？？