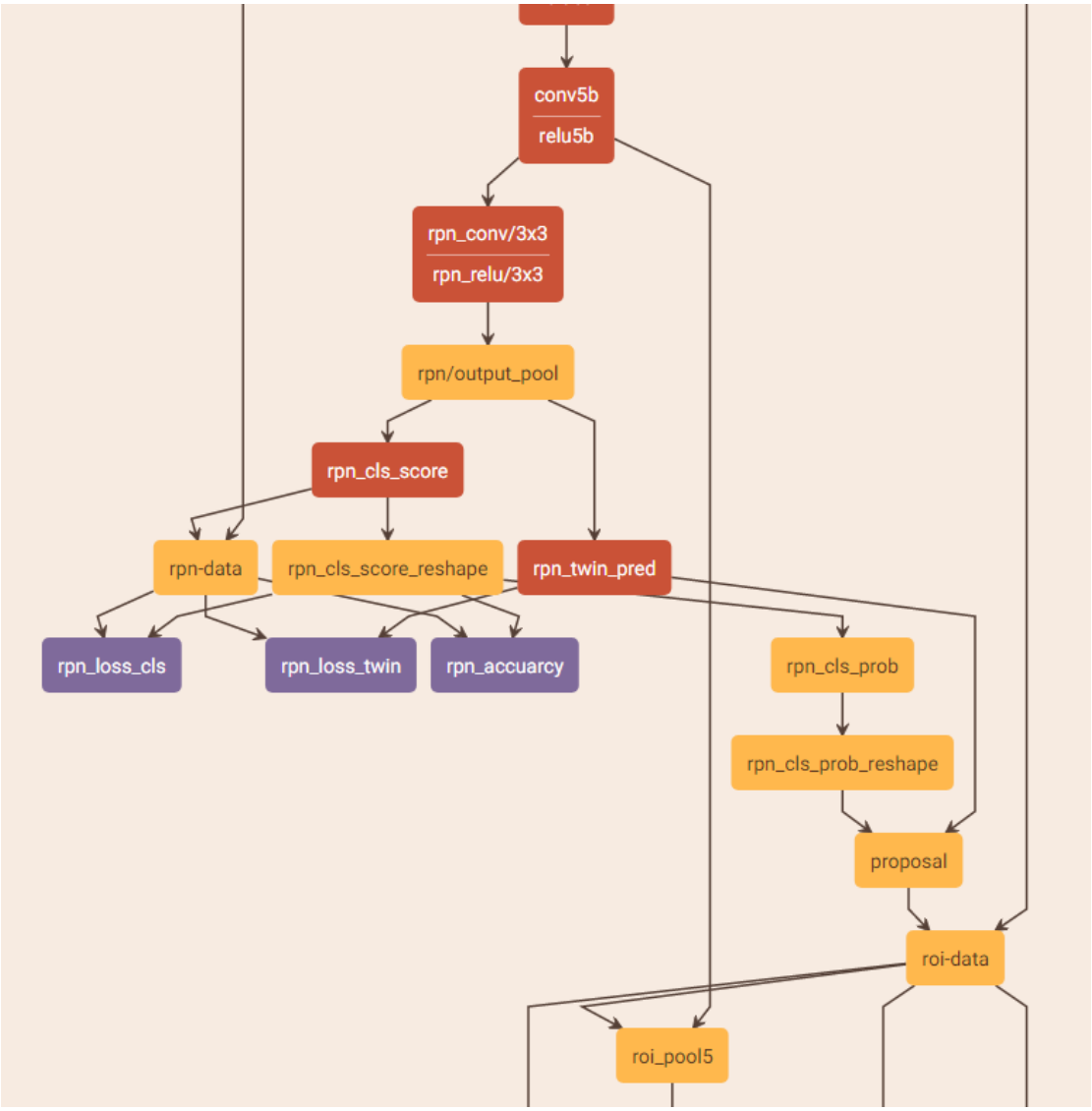
本周主要看了 faster rcnn 和 R-C3D 的部分代码并开始自己写 pytorch 框架下的 R-C3D 代码。从两份代码的 train 部分开始看，然后把 RPN 部分相关的代码理解并写了 pytorch 所用的 torch.nn.module 的子类的代码，此部分网络结构如下图所示：



这部分结构中间的输入为经过 conv5b 的 feature maps，得到的 rpn_cla_score 通过 reshape（channel 层变为 2，H 变为(H*W/2)）来判断正负样本，rpn_cls_prob_reshape 将输出的 channel 层变为 K+1 维作为 dectection 部分的类别置信度。rpn_twin_pred 也是通过 3 维卷积得到，C 层有 2*anchor 个，proposal 由 proposal_layer.py 得到。在训练时，通过交叉熵损失和 smooth l1 损失分别得到 class 和 anchor 得到的损失函数。torch.index_select(input,dim,index)中 index 指的是在 dim 维下的索引号，如第几行，列。

Proposal_layer.py:　ProposalLayer 完成的任务是生成 anchor，将 anchorclip 到对应视频位置，将预测滑框长度小于阈值的滑框删除，根据 score 排序，取前 N 有效个滑框，再对其做 NMS，将其结果传递给 roi-data。此处的 backward 不做梯度下降。这个地方由于是处理视频长度，是在特征的 D 维上进行处理，和 bounding box 在 H 和 W 上处理有差别，主要是要验证自己写的函数能否确保输入输出 shape 对的上。

目前正在改 anchor_target_layer 部分的代码，主要是用来做 proposal 模块的正负样本和划窗

的回归，此处他把所有 anchors 放在 all_anchors 里面进行处理，并删除了多余的正负样本保证正负样本比例，样本通过 np.random 采取了随机取样，里面很多处理是直接通过[::,]之类的方法处理的，用,来分割不同的维度，::是将一个维度的数据进行输出，:之间分别是开头，结尾，step，可省略。

目前的计划是下周先想办法把那份 R-C3D 的 caffe 代码跑出来，这周搭了一下 caffe 的环境，能知道中间一些输出的格式，同时也作为模式识别课的一份报告，由于下个月要开始有结课考试了，在月底前我尽量保证能把自己写的代码跑通。