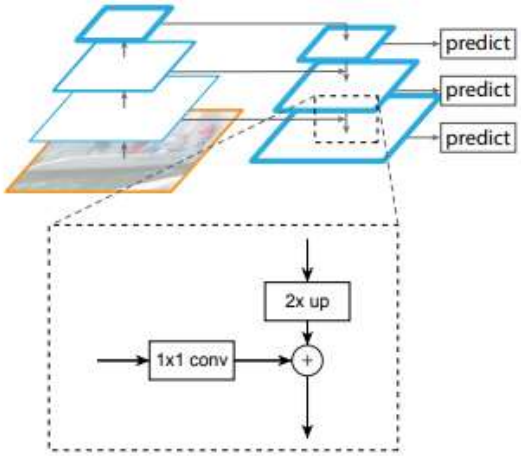
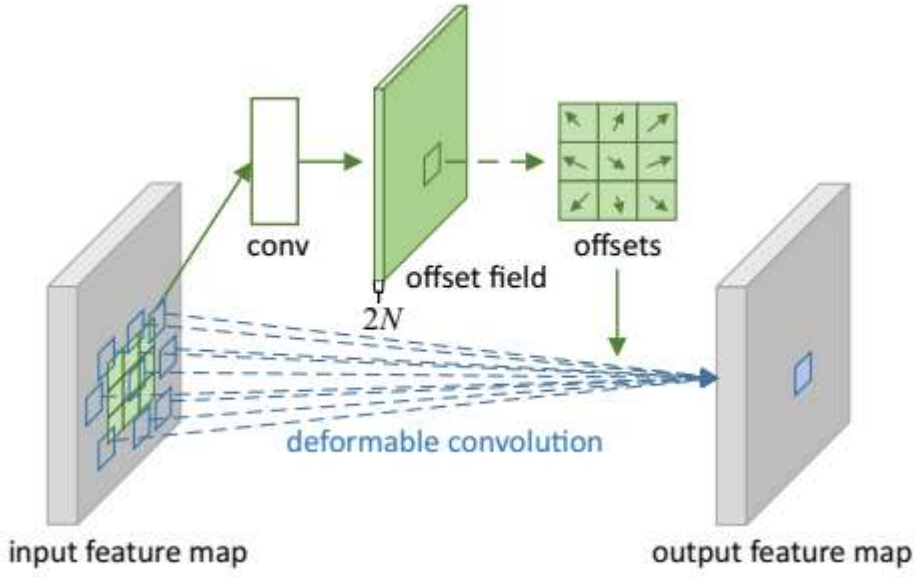
上学期期末把 R-C3D 用 resnet3D50 的预训练网络跑了一下，得到的 mAP 在 40 到 70epoch 的时候在 33、34 左右有浮动，再后面就有明显下降了。这学期开学有个课设，所以没在这方面花很多时间，主要把 activitynet 提帧和光流的程序搞好了，目前 rgb 已经提完，要清理下硬盘空间再继续提光流。在 check 的时候有些用 CV_CAP_PROP_FRAME_COUNT 得到的帧数和提到的图片数有区别，后来查到原因是 CV_CAP_PROP_FRAME_COUNT 给出的数据不一定百分之百准确，如果视频不给出这个值就会算一个大概值，因此会有一定的误差（https://stackoverflow.com/questions/31472155/python-opencv-cv2-cv-cv-cap-prop-frame-count-get-wrong-numbers）。

在看那两篇论文的时候其中有些网络和知识点没看过，就去看了些相关文章。在一篇博客上看了 FPN 的大致结构（https://blog.csdn.net/u014380165/article/details/72890275/），主要思路是顶层特征通过上采样和低层特征做融合，而且每层都独立预测。在代码里面的思路是每次 poling 后的层作为不同层的特征，左边的每个 feature map 通过 1*1 的卷积全部降成 256 维，右边的高层 feature map 会先通过 upsample 在 size 上增大，在和左边传过来的 feature map 相加，再经过一个 3*3 的卷积层得到下一层的 feature map（右边的蓝色框），后面每个蓝色框都会做后续的预测。



后来看了 Deformable convolutional network，这个用在了 trident network 的对比实验中，是为了验证两种对感受野的处理方法兼容，于是了解了一下。



主要的思想是对卷积的计算不单是在一个方框内进行，而是卷积求和的时候针对的每个点是通过另一个卷积来计算的，同时他也用类似的办法对 roi pooling 层进行修改，对每个要 max pool 的区域进行区分。offsets 的计算通过双线性插值来得到整数点，这样才确保了每个 feature map 整个的形状和原来的网络一样，但具体的 offset 计算方法还没搞透彻，他的反向传播的时候的计算流程在论文的最后也有推导，这一块准备找到合适的代码后再看清楚。

Trident network 文章当时没有想通他在 backbone 的中间几层用这个模块来做对比实验，后来想他意思应该是后续也是要分成三路，但每路后续不会使用空洞卷积，由于已经改变了感受野了，在中途没有办法融合，只能将得到的结果进行 NMS。第二个问题 weight share 有效作者说是因为三个 branch 对图像的处理只有尺寸上的区别而特征提取的流程是相似的，但在反向传播的时候由于共享最后又会对其他 branch 的参数产生影响，分析这个地方应该是由于反向传播是针对合适的 scale 那一 branch 的，所以在 loss 变小以后对整体的影响就变小了，最后的推论是针对三个输出的和，只要某 branch 的置信度高，那么最后得到的结果就会更好。