关于8XFCN的代码试验与心得

一、对FCN的理解

本次作业的FCN网络分为3个部分，第一部分采用VGG作为特征抽提的工具，第二部分采用双线性插值上采样、随机场作为将特征还原成与原图一样大尺寸的特征图，第三部分对每个像素的分类向量做一个分类器。

对于VGG\_16网络，其中共有5层pooling，若输入一个尺寸为224\*224的图片，则其会在第五层pooling后会输出7\*7\*512的特征图。同理在第四层pooling后输出14\*14尺寸的特征图，在第三层pooling后输出28\*28尺寸的特征图。

对于8XFCN网络，意味着最后将特征图进行8倍的上采样，此时输入上采样的特征图尺寸是28\*28\*n，此通过FCN的skip layler实现。先将第4层pooling输出的1倍上采样+第5层pooling输出的2倍上采样进行加和再做2倍上采样，然后再与VGG中的第3层pooling的输出进行加和。最终将得到的特征图再进行8倍上采样后变成50176\*21维的onehot向量与groundtruth进行比较。

代码实现：

1、先取出VGG中的第3层pooling的输出进行1\*1卷积得到28\*28\*21的特征图(以便后面调用)：

pool3\_feature = end\_points['vgg\_16/pool3']

with tf.variable\_scope('vgg\_16/fc8'):

aux\_logits\_8s = slim.conv2d(pool3\_feature, number\_of\_classes, [1, 1],

activation\_fn=None,

weights\_initializer=tf.zeros\_initializer,

scope='conv\_pool3')

2、第4层pooling输出进行1\*1卷积得到14\*14\*21：

pool4\_feature = end\_points['vgg\_16/pool4']

with tf.variable\_scope('vgg\_16/fc8'):

aux\_logits\_16s = slim.conv2d(pool4\_feature, number\_of\_classes, [1, 1],

activation\_fn=None,

weights\_initializer=tf.zeros\_initializer,

scope='conv\_pool4')

3、第5层pooling输出的2倍上采，得到14\*14\*21：

upsampled\_logits = tf.nn.conv2d\_transpose(logits, upsample\_filter\_tensor\_x2, output\_shape=tf.shape(aux\_logits\_16s), strides=[1, 2, 2, 1], padding='SAME')

4、将第4、5层数据加在一起形成14\*14\*21的特征图

upsampled\_logits = upsampled\_logits + aux\_logits\_16s

5、对得到的特征图再做2倍上采样，得到28\*28\*21特征图：

upsample\_filter\_np\_x2x2 = bilinear\_upsample\_weights(2, number\_of\_classes)

upsample\_filter\_tensor\_x2x2=tf.Variable(upsample\_filter\_np\_x2x2,name='vgg\_16/fc8/t\_conv\_x4')

upsampled\_logits=tf.nn.conv2d\_transpose(upsampled\_logits,upsample\_filter\_tensor\_x2x2, output\_shape=tf.shape(aux\_logits\_8s), strides=[1, 2, 2, 1], padding='SAME')

6、将其再与pooling3输出的特征图进行相加：

upsampled\_logits = upsampled\_logits + aux\_logits\_8s

7、再做一个8倍的上采样，以得到224\*224的特征图，供展成onehot编码与groundtruth比对：

upsample\_filter\_tensor\_x8=tf.Variable(upsample\_filter\_np\_x8,name='vgg\_16/fc8/t\_conv\_x8')

upsampled\_logits=tf.nn.conv2d\_transpose(upsampled\_logits,upsample\_filter\_tensor\_x8,output\_shape=upsampled\_logits\_shape,strides=[1,upsample\_factor, upsample\_factor, 1], padding='SAME')

二、心得：

1、本周作业遇到的第一个小问题是没有注意模型的相关参数，导致出错。

2、由于网速问题，再安装opencv-python后多次报错折腾了很久。

3、对于FCN中的跳接仍有个疑问，为什么不一次性对pooling3\*1，pooling4\*2，pooling5\*4进行加和，而是将pooling3\*1+（pooling4\*1+ pooling5\*2）\*2。这样做是为了减少计算两，还是确实效果更好？