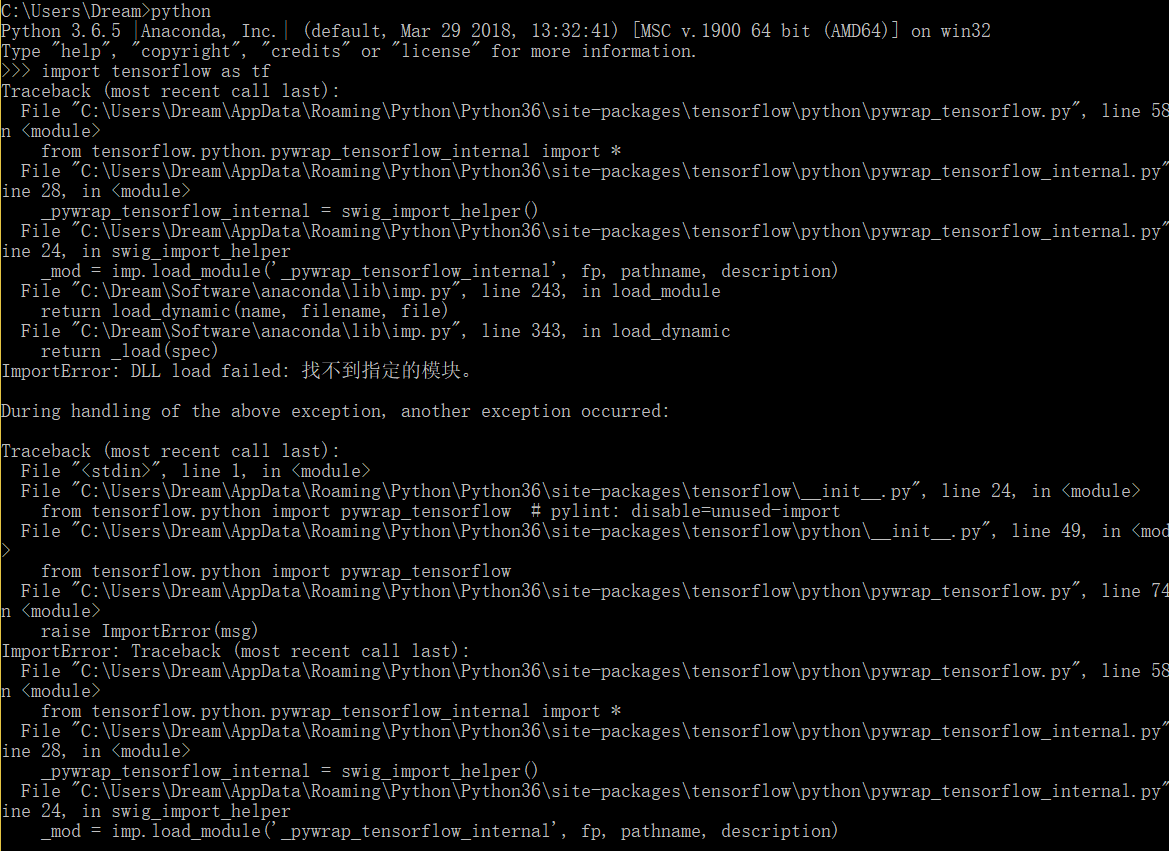
问题1：tensorflow安装失败报错Could not install packages due to an EnvironmentError:

[WinError 5] 拒绝访问

解决方案：改为pip install --user --ignore-installed --upgrade tensorflow-gpu

问题2：安装tensorflow后使用时报错



解决方法：

<https://blog.csdn.net/u012011332/article/details/84102986>

tensorflow新版本不支持cuda9.1，使用网友编译好的版本安装即解决问题

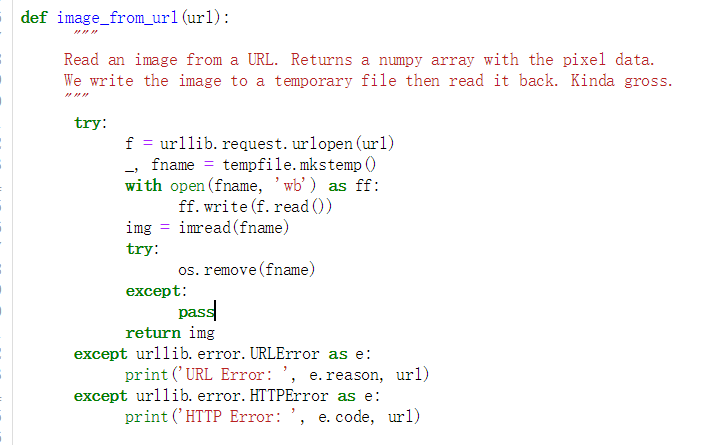
问题3：用提供的.sh文件下载不了数据

解决办法：用记事本打开几个点.sh，直接复制里面的网址信息下载

问题4：运行时候报错



解决方法：查找资料后将imagee\_utils中函数image\_from\_url修改为如下



问题5：

LSTM\_Captioning中，运行evaluate\_model()时报错

1. NameError: name 'med\_data' is not defined cs231n

解决：改为minibatch = sample\_coco\_minibatch(small\_data, split=split, batch\_size=1000)

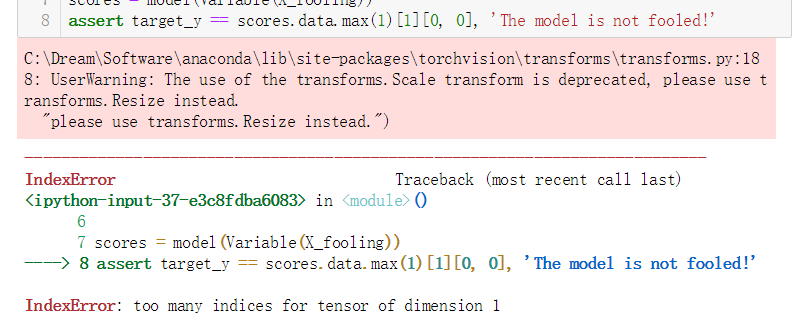
1. NameError: name 'BLEUscores is not defined cs231n

解决：预先定义下BLEUscores={}

1. NameError: name nltk is not defined cs231n

解决：import nltk（Natural Language Toolkit，自然语言处理工具包，在NLP领域中，最常使用的一个Python库）

问题6：Q3中 fooling map报错



[1][0,0]改为[1][0]

查看cuda版本：

Nvcc -V

我的cudnn9.1, cudnn7.4

Microsoft COCO 数据集

COCO数据集是微软团队获取的一个可以用来图像recognition+segmentation+captioning 数据集。

The dataset consists of 80,000 training images and 40,000 validation images, each annotated with 5 captions.

SqueezeNet

使用已经抽取出的特征并降维到512维。使用int代替string，因为string相对来说难以处理。保存int-string的映射。

加了一些特殊tokens，比如开头加`<START>`，结尾加`<END>`。Minbatch中可能有长短不同的caption。，将短caption用`<NULL>`补齐，在计算Loss和梯度时，不计算`<NULL>`的梯度。特殊token已经处理好了。

tanh(x) 的导数是 (1−tanh2(x))

range(start, stop[, step]) -> range object，根据start与stop指定的范围以及step设定的步长，生成一个序列。

db=np.sum(dout,axis=0)#axis=0表示按行求和

cache=[]#cache应该是元组的列表,且大小为T。D=cache[0][0].shape[1]#cache为长度为T的元组列表，cache[0]表示列表中第0个元组，[0][0]表示元组中第0个元素也就是x

Numpy切片

:表示该维度取所有元素

X[:,0]：取X第0列的所有元素

X[1,:]:取X第一行所有元素

X[:5]：表示从0开始取到第四个元素

X[1:5]:表示从1开始取到第四个元素

X[1:-1:2]:表示从第一个开始直到末尾每个一个元素取一个出来，2为步长，可以为负数，表示倒着取，此时开始下标必须大于结束下标。

X[::-1]:表示从尾到头取所有元素-1表示每次向左移动一个元素

inds是一个一维array，则x[inds]返回的是一个二维array；但是如果inds是一个二维array，则x[inds]返回的是一个三维array。

np.add.at(x, [0,2], 3) # 下标0和2的元素分别加3

Q1、

1. RNN单个单元的正向反向传播
2. RNN总体的正向反向传播注意点见笔记
3. Word embedding：把用数字表示的单词转换成数组

词表中有V个单词，每个单词用一个D维向量表示

输入时(N,T)维的数组每一行表示图片对应的一句有T个单词的caption，

其中每个单词用单词在词汇表中的下标Indice表示。还有一个权重矩阵W=(V,D),给出了词汇表中每个单词的向量表示

前向传播：out=W[X,:]，说实话这里不是很理解

1. temporal affine layer：每个隐藏层向量使用affine转化为每个单词的分数。

即，输入x(N,T,D)使用权重矩阵和偏差使之变为输出out(N,T,M).这个用来干嘛呢

1. temporal\_softmax\_loss:每一个时间戳都会得到一个当前的loss。在时间顺序下回累加各个时间的loss。由于不同caption单词个数不同，少的用null补充，并且维护一个mask表示当前生成的caption共几个单词，让NULL不参与损失函数的计算
2. Vanilla RNN

每个图用一个D维向量表示。单词表中共有V个单词，每张图输出T个单词。隐藏层维度为H。词向量维度为W

总输入N\*D，

W\_proj是什么东西

1. Vanilla RNN loss计算
2. 使用affine将输入图像特征features, N\*D通过W\_proj(D,H)权重矩阵变为初始隐藏状态h0(N\*H)
3. 使用word embedding(N\*T)将真正的ground truth单词（除去最后一个，大小为N\*T-1，此处之所以要不最后一个是因为rnn是根据前一个单词计算后一个单词的，句子总长为V，而第V个单词输入之后并不用输出V+1个）从Indices转换为向量（N\*T\*W，即x）。使用的权重参数为W\_embed（V\*W）
4. 使用rnn\_forward将x(N\*T\*D)和初始影藏层状态h0通过rnn传播，得到T个时间隐藏层状态h(N\*T\*H),使用权重为Wx,Wh
5. 使用temporal\_affine\_forward计算各个时间隐藏层h(N\*T\*H)状态对应的输出各个单词分数out(N\*T\*V)。作用是将隐藏层各变换为每个数据的每个时间每个单词对应的分数。即输入（N,T,H）变换为输出（N,T,V）
6. 使用temporal\_softmax\_loss计算每个隐藏层单词得分和ground-truth直接的softmax loss。输入为out，和captions\_out(不包含第一个单词)。这里为不要第一个，因为我们RNN生成的是后V-1个，这才需要算到Loss里面
7. 测试时，每次从单词表得分分布中采样最大的那个作为输出，并将其作为输入传入下一个RNN单元
8. 根据features计算初始隐藏状态prev\_h(N,H)
9. 设置初始单词的indice数组Prev\_word(N)
10. 对于第0到第t-1个时间，
11. 将prev\_word作为输入，使用word\_embed编辑为向量形式(N,W)x
12. 将向量形式的x输入单个rnnsetp，计算下一h（N,H）
13. 将h(N,H)大小变换为(N,1,H)（每次只有上一个单元的隐藏层数据）然后进行affine transformation得到V个单词对应的得分(N,1,V)，并取最大的作为真的单词输出，记录下

Q2: LSTM

1. **梯度消失和梯度爆炸**

RNN会丧失学习到连接很远的信息能力

**梯度消失原因：**

a.tanh激活函数取值范围是-1到1。

b.末尾的梯度[-1,1]之间的值，传到第一个时间单元的时候经过多次矩阵乘法[-1,1]之间的值，使得很多梯度已经几乎接近于0，从而导致了梯度消失

**梯度爆炸：**

当使用relu激活函数时，取值为[0,无穷]，多次乘法容易导致梯度爆炸。可以使用梯度裁剪解决（当梯度的范数大于某个给定值时，将梯度同比收缩）

Relu的导数是常数值0或者1，不会引起梯度消失

1. **为何LSTM能解决梯度消失**

Vanilia里面s(t) 是tanh(…)，链式法则求导是连乘的形式，只有有一个接近于0，总体梯度就接近于0。而lstm的ct=ft\*ct-1+it\*ct\_,导数值时相加的形式。

1. **如何训练一个能较好拟合验证集的网络**

可变参数：hidden\_dim，wordvec\_dim(单词向量的维度W)，num\_epochs, batch\_size, learning\_rate,lr\_decay

其中hidden\_dim，wordvec\_dim越大，网络结构越复杂

Num\_epochs越大，越能拟合数据

Batch\_size要合适，过大的话一次forward时间会过多

Learning\_rate要慢慢调

Lr\_decay看学习情况

Test1:

Batch\_size=64,完成所有数据的前向要32次(batch\_size的推荐值是32，64，128)

Print\_every=100次（大概进行了三次前向），输出loss，看Loss有没有下降：77->30

如果变大lr:5e-3->1e-2 loss:77->27->16看起来更好了一点

Test2:

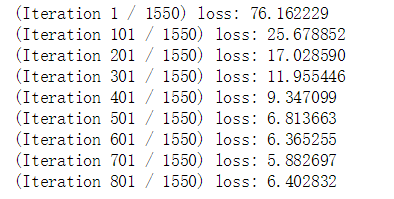
保持test1数据

增大wordvec\_dim=512 loss:76->25

Test3:

保持test1,2设定，使hidden\_dim=32

decay越大，学习率衰减地越快



Lr\_rate=1e-2, lr\_decay=0.995出现震荡

题目的要求是不要过拟合，要让模型在验证集上也跑的很好，所以Hidden\_dim和wordvec\_dim感觉应该不要太大

Test4:

Hidden\_dim=32,lr=2e-2

**Q3: Network Visualization: Saliency maps, Class Visualization, and Fooling Images**

在预训练好的网络上定义关于图片好不好的loss函数，用后向传播计算loss对于图像各像素点的梯度

实验设置：

1. 实验给出的是经过归一化处理的图像。
2. 使用在Imagenet上训练好的squeezeNet
3. **saliency map**

saliency map即特征图，可以告诉我们图像中每个像素点对图像分类得分的影响。

计算它的时候首先要计算图像每个像素对应的正确分类中的标准化分数的梯度(这是一个标量)。如果图像的形状是(3, H, W)，这个梯度的形状也是(3, H, W)；对于图像中的每个像素点，这个梯度告诉我们当像素点发生轻微改变时，正确分类分数变化的幅度。计算saliency map的时候，需要计算出梯度的绝对值，然后再取三个颜色通道的最大值；因此最后的saliency map的形状是(H, W)为一个通道的灰度图。

先跑model得到每张图对应每个类别的各个得分，取正确类别对应的分数为每张图loss（N,）后向传播计算图的梯度。取绝对值以及三通道最大值作为图的梯度.梯度很大的地方就是影响强烈的地方。

Pytorch:

a.gather函数: torch.gather(input, dim, index, out=None) → Tensor或者input.gather(dim, index, out=None)

沿给定轴dim，将输入索引张量index指定位置的值进行聚合。如s.gather(1, y.view(-1, 1)).对于s的第一维，以y为索引去选择元素。具体可以见“Hint: PyTorch gather method”

b. squeeze(): 去除size为1的维度，包括行和列。如将4\*3\*1的向量变为4\*3维

c.pytorch backward()函数。如果 backward() 没有参数，调用 backward() 函数的变量必须是一个标量，即形状为(1,)，否则就会报错。如果 backward() 有参数，比如 backward(torch.FloatTensor([0.1, 1.0, 0.001]))，那么此时调用 backward() 函数的变量可以不是标量，也可以是向量，但是要注意的是，这个向量的维度必须和 backward() 的参数向量的维度相同，其实此时就相当于函数有多个输出，每个输出都要算一个梯度，并且每个输出算出来的梯度的步长对应于 backward() 的参数向量对应的维度，且最终某个输入 x 的梯度是所有这些输出针对 x 算出的梯度的加权求和(权值向量就是 backward 的参数向量)。

d. 在BP的时候，pytorch是将Variable的梯度放在Variable对象中的，我们随时都可以使用Variable.grad得到对应Variable的grad。如要得到x\_var的梯度，可以直接用x\_var.grad.

c. python切片列表切片，a[起始指标 : 终止指标：跳跃步长]。这三个指标的意思就是从哪里跳到哪里，每次跳几个。如果第三个没有，默认为1

d. torch.cat([right, left], dim=3).如果第二个参数是1，torch.cat就是将a,b 按列放在一起，大小为torch.Size([1,4])。如果第二个参数是0，则按行。此处参数为3应该让第三个维度拼接

行放在一起

1. **fooling images**

顾名思义，就是指一张图片，虽然上面通过肉眼看到的是松鼠（举个例子），但是因为这张图片加了一些特定的噪声，所以神经网络会将它误识别为其他物体。

对每张图进行处理：先跑Model，然后看其得分最大的是否为需要调整的类别，若是退出循环，返回图片；否则，求图对应梯度并使用梯度上升更新图片，直至产生目标类别的图片。

为什么是梯度上升而不是梯度下降：梯度下降是求最小值，梯度下降是求最大值。本题选用的loss是target\_y的分数，目标是使分数更高，所以应该梯度上升。（把图像看出输入，中间复杂网络看出是计算图中间内容，输出是target\_y分数，要使分数变大，后向传播梯度到图的时候，应该使用加法梯度上升）

**Q4: Style Transfer**

使用在imagenet上预训练过的squeeze net。对图梯度下降而不是参数权重

使用一个同时匹配content image和style image的图。Loss是weighted sum of three terms: content loss + style loss + total variation loss。

***本来想用pytorch，后来因为对pytorch实在不熟，网上又没找到pytorch的版本参考。为了节省时间，最后选择用tensorflow***

1. content loss

将每一层的特征图转化为向量，与source map的向量求范数

B. Style Loss

用gram矩阵近似每个filter之间的协方差，希望我们生成图像的激活统计数据和style image的激活统计数据近似，匹配协方差就是一种可以达到这种效果的方法。实际中会在几层之后将生成图和source图的gram矩阵求范数

C. total-variation正则化

使图片平滑。使用三通道上各像素差值

图像是从内容图像开始，使用loss逐步使其余内容图像内容符，和风格图像风格同，并减少模糊

其他应用：

从一张噪声图像开始，将style loss 权重设置为0，可以生成一张和content image非常近似的图片。

同理将content loss 权重设置为0，可以生成style image纹理的图片

Tensorflow

1. tf.reduce\_sum（）//如果不指定第二个参数，那么就在所有的元素中取平均值
2. tf.matmul（）//矩阵乘法，将矩阵a 乘于 矩阵b。输入必须是矩阵（或者是张量秩 >２的张量，表示成批的矩阵），并且其在转置之后有相匹配的矩阵尺寸。
3. tf.cast( x, dtype, name=None)将x的数据格式转化成dtype
4. tf.nn.sigmoid\_corss\_entropy\_with\_logits

函数的作用就是计算最后一层是sigmoid层的cross entropy，只不过tensorflow把sigmoid计算与cross entropy计算放到一起了，用一个函数来实现。输入**logits(sigmoid(logits)为网络输出)**，它的shape是[batch\_size, num\_classes] ，一般来讲，就是神经网络最后一层的输入z。另外一个输入是**labels(y)**，它的shape也是[batch\_size, num\_classes]，就是我们神经网络期望的输出。logits: 一个事件发生与该事件不发生的比值的对数，logit 代表的意思 并不是 logit 的数学含义。 当成 未归一化的概率 就好。output不是一个数，而是一个batch中每个样本的loss,所以一般配合tf.reduce\_mea(loss)使用。

5、tf.reduce\_mean 函数用于计算张量tensor沿着指定的数轴（tensor的某一维度）上的的平均值，主要用作降维或者计算tensor（图像）的平均值。

**Q5. : Generative Adversarial Networks**

更新目标：更新G来最大化D在生成数据上作出正确判断的可能性；

更新D来最大化D在真实数据和生成数据上正确判断的可能性。

数据集：MNIST

D网络结构：全连接-leakyRelu-全连接-leakyRelu-全连接

优化函数：AdamOptimizer with a 1e-3 learning rate, beta1=0.5 to mininize G\_loss and D\_loss separately.

Least Squares GAN:对损失函数做了优化，更加稳定

DCGAN：使用卷积，会有更多边缘信息

D网络结构：conv-leaky relu-maxpool-conv-leaky relu-maxpool-flatten-全连接-leaky relu-全连接

G网络结构：全连接-relu-bn-全连接-relu-bn-resize into image tensor-卷积-relu-bn-conv-tanh