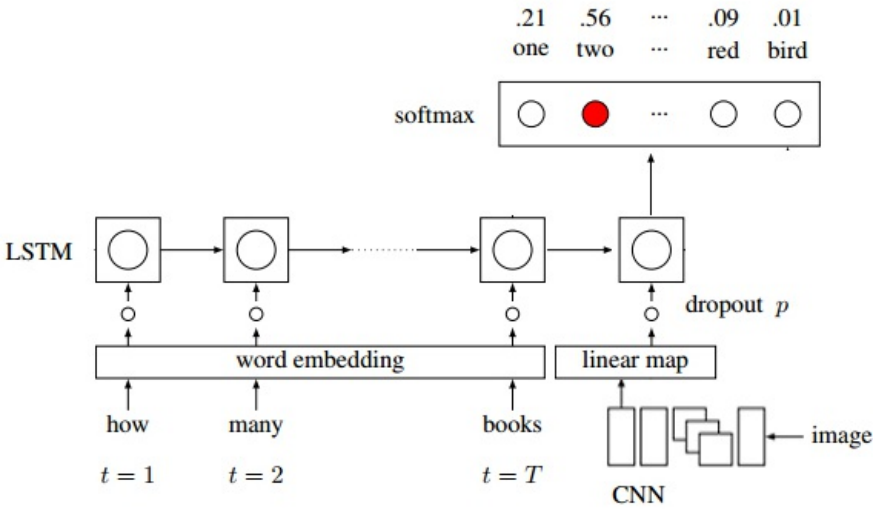
VQA任务报告

baseline的测评

baseline简单介绍

baseline可以解决VQA-v1和VQA-v2问题，用的是tensorflow框架，可以在验证集上得到50%左右的正确率
注：baseline原本要求python2.7，我改了它的代码让它可以在python3.6上跑
baseline地址：
https://github.com/paarthneekhara/neural-vqa-tensorflow
网络结构：



- baseline中image embedding用的是已经训练好的VGG16模型（提取fc7层的特征）
- LSTM的层数默认为2
- github中附带的baseline的论文链接似乎已经失效，这里补充一个链接
  http://papers.nips.cc/paper/5640-exploring-models-and-data-for-image-question-answering

数据集的使用方法

注：我利用baseline解决的是VQA-v1数据集的MultipleChoice任务，并未涉及到VQA-v2的OpenEnded任务
将相应的图片和数据放到本地的Data文件夹（Data文件夹和代码同一级）后，首先执行`data_loader.py`中的`prepare_training_data`

`prepare_training_data`用来处理数据集中的json文件（即训练集和验证集的问题和答案），进行了如下操作：
* 建立语料库（统计问题和答案中词汇的词频）
* 根据之前统计的词频，将每个图片对应的问题和答案转化成词向量
* 保存上述两步得到的字典（保存到qa*data*file1.pkl和vocab_file1.pkl中）

接着运行`extract_fc7.py`，这个py文件将验证集和训练集的图片(先转换成(224, 224, 3)的形状后)放到已经训练好的VGG16模型中，得到**fc7层**的特征并保存下来（保存在train_fc7.h5中）

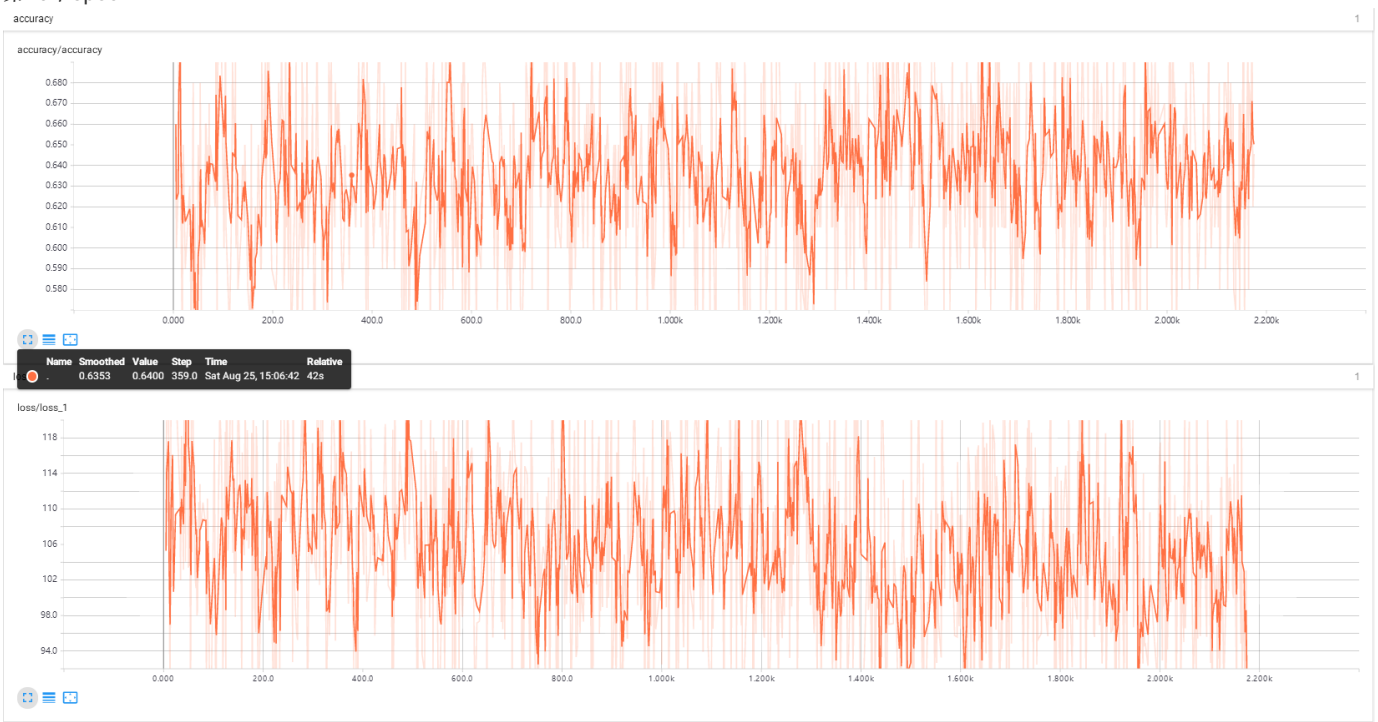
- 注1：baseline的github中给出了训练好的VGG16模型的下载地址
- 注2：extract*fc7.py还保存了之后用于训练的图片的id（保存在train*image*id*list.h5中）
- 注3：VGG16的论文链接：http://arxiv.org/abs/1409.1556.pdf

TensorBoard可视化

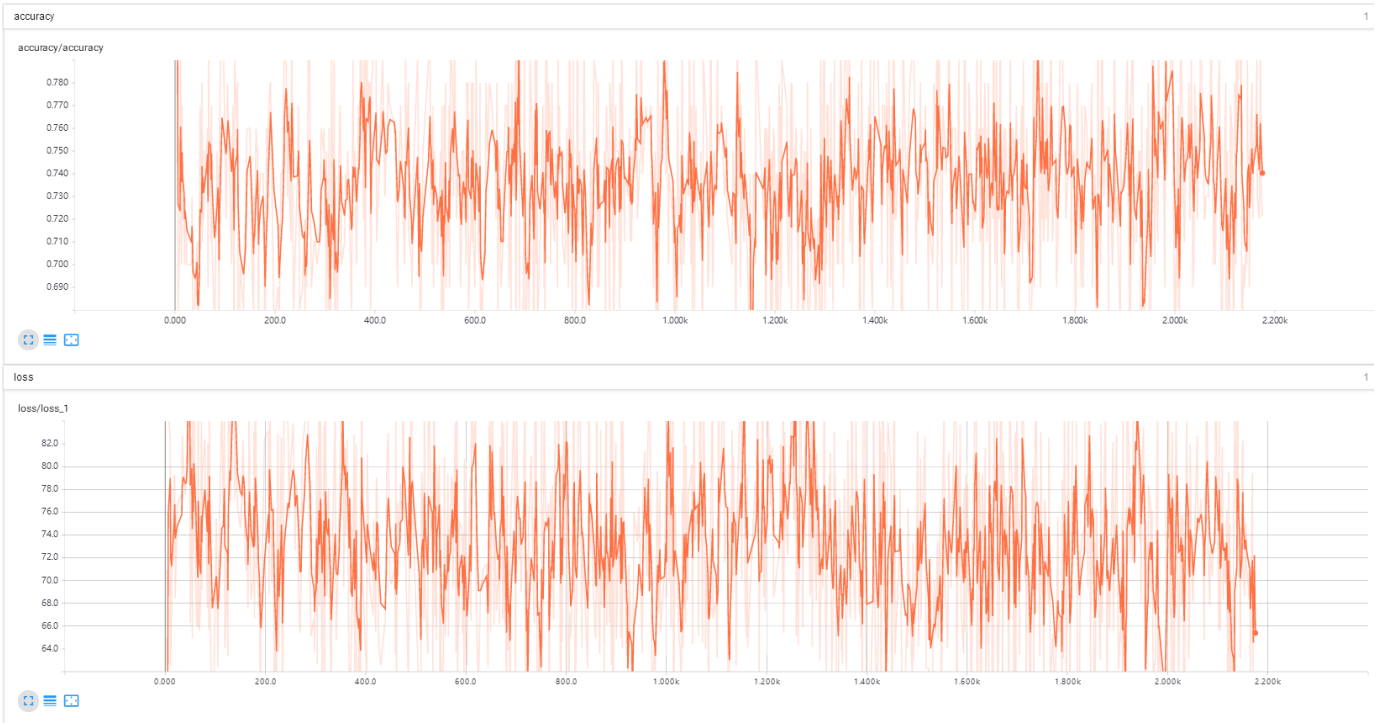
完成数据集处理后，按照train.py的默认参数进行训练20次（根据baseline的github上的描述，训练12个epochs就已经有对验证集的最好效果了）
在原有代码加上对Loss和训练集accuracy的可视化，训练过程中每一个epoch后就保存一次logs
注：由于在最开始的baseline的可视化logs已丢失，这里只能放出batch_size为100的结果（默认为200）

- 第1个epoch:

- 第10个epoch:



- 第20个epoch:



测评结果

在验证集中进行测评（运行evaluate.py），得到的正确率约为**0.47**
注：在未做修改的baseline上的测评结果（截图等）不小心丢失了，后续对模型的改进部分会放出相应的测评结果截图

baseline的改进

1、直接在baseline上调参并微调模型
初步进行了如下调参（参考了网上一些训练trick）： * batch*size从200改成了100，并相应地减少了epochs*
* *将image*embedding部分的激活函数从tanh改成了relu
* 将LSTM的层数从2改成了3

2、对word embedding部分进行了改进
主要进行了如下改进： * 对每个图片对应的问题的词汇进行了词性还原（用到了zltk框架，同时把be动词全部用'be'来代替）
* 去掉了问题中一些不必要的词汇（例如'to'、'a'、'an'等）

完成了上述两步修改后，训练了20个epochs，对得到的新模型进行测评，测评结果如图：

```
1059 Acc: 0.420000
1060 Acc: 0.600000
Total Acc 0.49161320404624037
```

注：选用第17个epochs得到的model是因为之后的epoch得到的model效果都变差了

可以看出对于未修改的baseline，修改后的代码正确率确实有所提升，我认为原因在于以下几点：

- 将batch_size降低可以用更小的epochs训练出效果更高的模型
- 对于image_embedding而言relu作为激活函数比tanh更好
- 进行词性还原、去掉不必要的词汇，在不改变问题原意（或改变较少）的同时，降低了语法和不必要的词汇对机器理解问题原意的影响
  注：提升LSTM的层数后经过对比，发现LSTM是2层的时候模型效果更好，所以这个改进不可取

3、改变image embedding的模型
将image embedding用到的`VGG16`模型改成了`ResNet101`和`Inception v3`
由于用的是tensorflow框架，我直接用tensorflow的slim中训练好的模型进行评测，同时仿照baseline的做法提取并保存相应特征
- **ResNet101**：提取最后一个全连接层的特征（shape为(1, 2048)），论文链接：https://arxiv.org/abs/1512.03385
- **Inception v3**：提取最后一个dropout层的特征（shape为(1, 2048)，keep_prob设为1），论文链接：http://arxiv.org/abs/1512.00567

之后进行训练，epochs设为10（经过多次对比可知当`batch_size`为100时，训练10个epochs即能得到较好效果），得到的结果如下：
- **ResNet101**:

```
1060 Acc: 0.400000
Total Acc 0.5237924492865239
```

- **Inception_v3**:

```
1060 Acc: 0.400000
Total Acc 0.5061320728288506
```

- **VGG16**:

```
1060 Acc: 0.400000
Total Acc 0.5073018833432558
```

根据上述结果可知，选用ResNet101作为image embedding的模型，效果要比VGG16好

4.还可以改进的部分

在参考了https://arxiv.org/abs/1707.04968v2 和另外几篇论文和博客之后，发现可以对baseline的网络结构进行更改，如加入`Augmented memory`等。但由于时间关系，这部分工作并未完成。

测试样例截图