Encoder and Attention实施

# 1.Encoder and Attention探索

根据查找到的一些资料及论文显示，encoder部分使用Inception-V4和Inception-Resnet-V2的效果比基础im2txt中使用的Inception-V3的效果更好。增加了Attention机制的模型比基础im2txt预测结果更好，因此考虑在我们项目中进行这两部分尝试。思路如下：

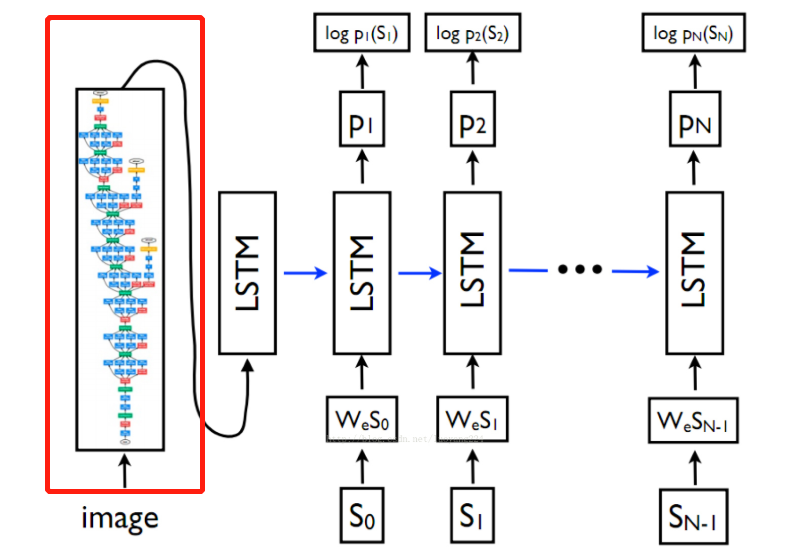
首先，研究并运行了基础的im2txt框架代码，观察了基础im2txt的loss情况，收敛速度以及预测情况。

其次，先进行了encoder部分的替换。分别使用Inception-V4和Inception-Resnet-V2替换im2txt中的Inception-V3，观察了loss情况，收敛速度以及预测情况

最后，研究了attention机制并将attention加入到im2txt框架下，分别调试加入attetion后的train，evaluate，inference。在这个过程中，需要先了解Inception-V4和Inception-Resnet-V2的网络结构，找出可作为输入的卷积层。最重要的是如何改造基础im2txt中的dynamic rnn部分，将attention layer融合进去。

## Show and Attention 基础模型

下图是show and attention的基础模型，左侧红框部分是encoder部分，也是本次项目中要尝试用inception-v4和inception-resnet-v2实现的部分。右侧是使用lstm的decoder部分，本次项目这部分也要改造，增加上attention layer机制。

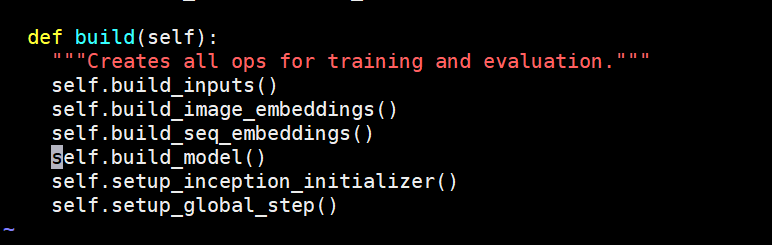


## Im2txt代码结构介绍

进行encoder部分的替换之前，需要对对im2txt代码结构有一个清晰的认识，才能准确进行修改，因此首先对im2txt的代码结构、重点文件和重点函数进行简要介绍：



重点说一下show\_and\_tell\_model.py的结构，该文件的结构也非常清晰，如下图所示，按顺序简要说明每个函数的功能：



* build\_inputs( )是根据读取的tf-record文件，利用图片信息和标注信息，对输入信息做处理，输出后续需要使用的image信息（预处理后的图像信息），input-seqs信息（输入的词序列），target-seqs（输出的词序列），input-mask（词序列的掩码）
* build\_image\_embeddings( )将预处理后的图像信息，输入所使用的特征提取模型，返回提取后的特征，用于rnn的输入
* build\_seq\_embeddings( )利用课程中学过的word2vector，将输入的每个词转换成vocabulary中对应的词向量，用rnn的输入
* build\_model( )创建rnn模型，利用了BasicLSTMCell以及dynamic rnn实现模型，包括train\evaluate\inference部分。输入是由前面两个函数提供的输入。
* setup\_inception\_initializer( )用来读取inception部分的checkpoint文件，恢复inception部分模型的参数
* setup\_global\_step( )创建global step的tensor

# 2.Encoder部分的替换

经过上面的对代码的分析，对encoder部分图像特征提取部分进行替换，需要改动2个文件和调用新的inception模型文件。

改动的文件是：

* show\_and\_tell\_model.py
* ops/image\_embedding.py

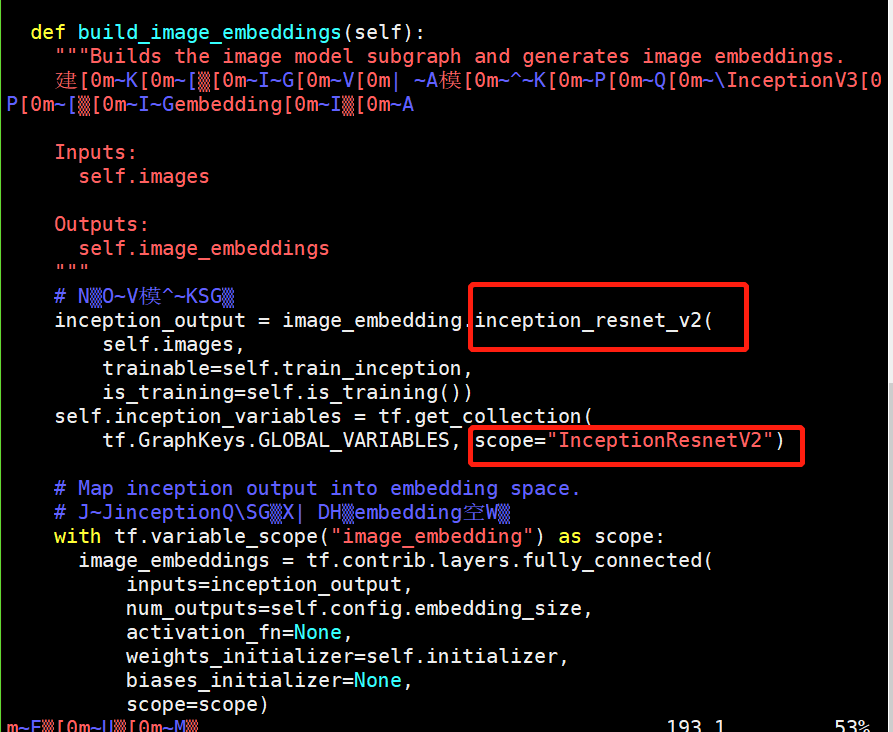
调用的新的inception模型文件是：

* slim.nets.inception\_v4
* slim.nets.inception\_resnet\_v2

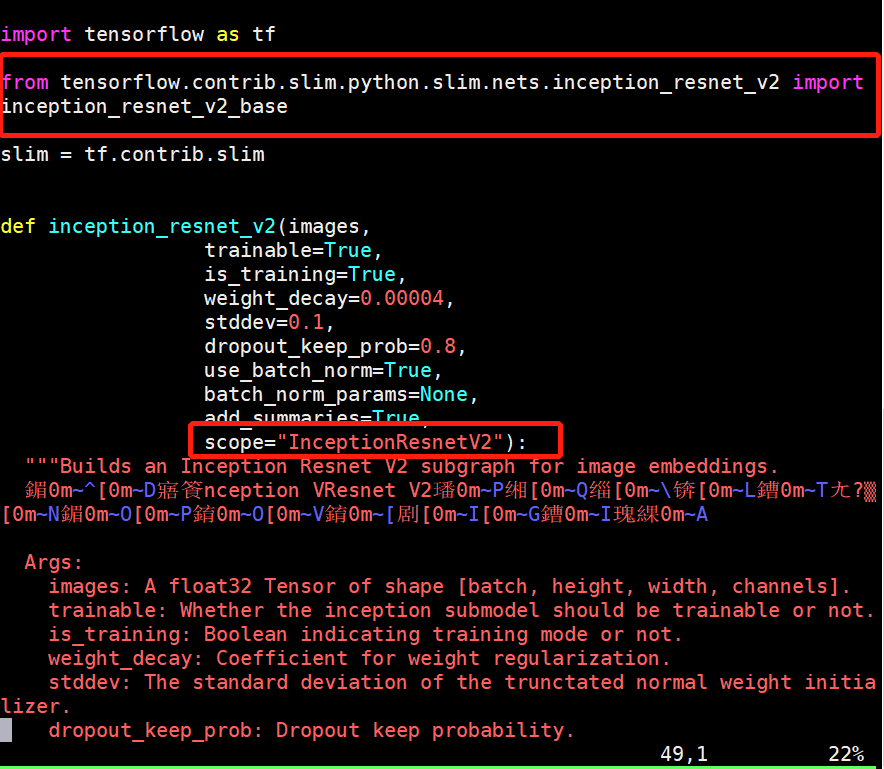
## 2.1 Encoder使用Inception-v4 和 Inception-Resnet-v2

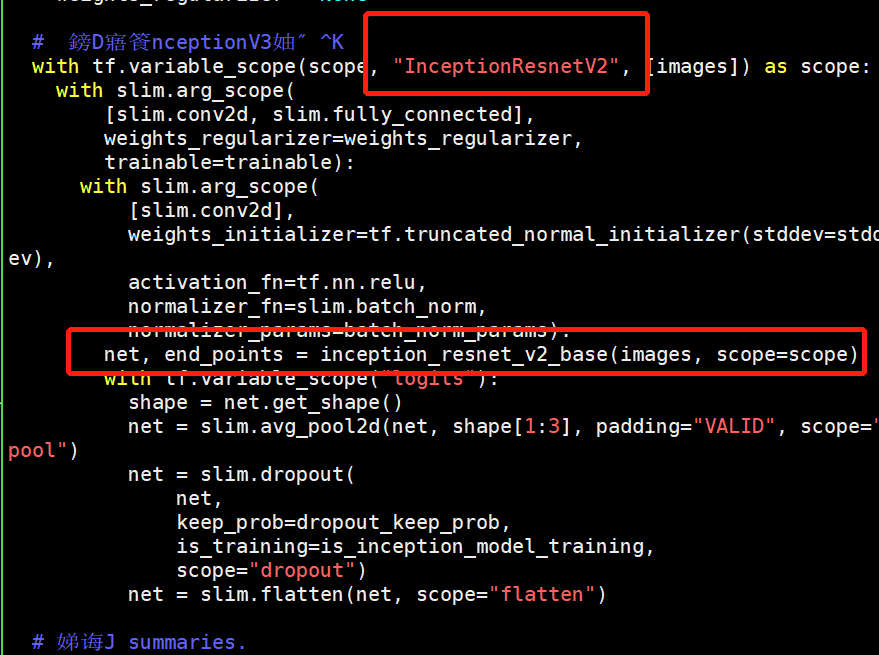
Inception-v4 和 Inception-Resnet-v2 代码修改部分基本一致，以下是以Inception-Resnet-v2为例

* show\_and\_tell\_model.py中主要修改build\_image\_embeddings( )函数，如下图中红框所示，将inception\_v3替换为inception\_resnet\_v2，注意同时修改scope的name，避免混淆。



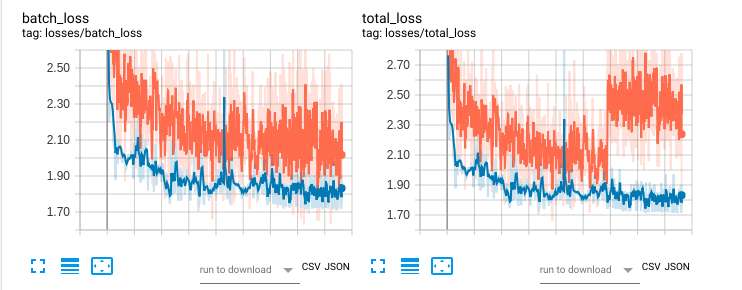
* ops/image\_embedding.py主要修改下图所示的位置，也是将inception\_v3的部分替换为inception\_resnet\_v2，同时注意scope name的替换，避免混淆





## 2.2 Inception-v4与Inception-v3的比较

Inception-V4 loss收敛图：（200w step）



演示图片（1）：

训练过程中多次试验此图片，发现每次将图片里trump判定为女性的概率要远远大于为男性的概率



* a man in a suit and tie standing in front of a building. (p=0.000287)
* a man in a suit and tie standing next to a woman. (p=0.000272)
* a man in a suit and tie standing in front of a store. (p=0.000090)

演示图片（2）：



Inception-v3 caption：

* a baseball player swinging a bat at a ball (p=0.003555)
* a baseball player swinging a bat at a ball. (p=0.001912)
* a baseball player holding a bat on a field. (p=0.001404)

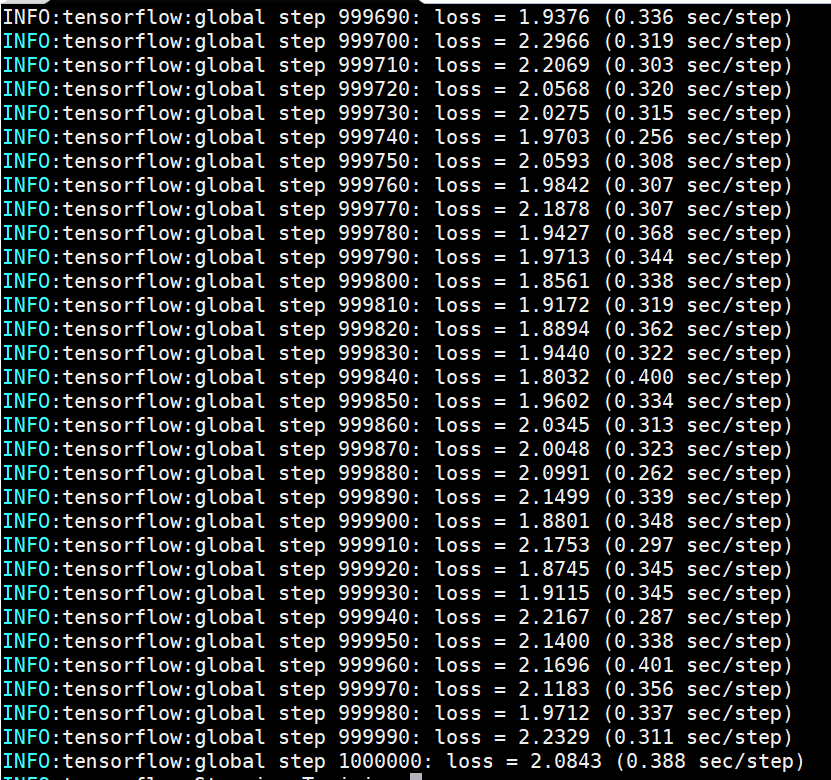
Inception-v4 caption：

* a young boy swinging a baseball bat at a ball. (p=0.002606)
* a baseball player swinging a bat at a ball. (p=0.002268)
* a young boy swinging a baseball bat on a field. (p=0.001090)

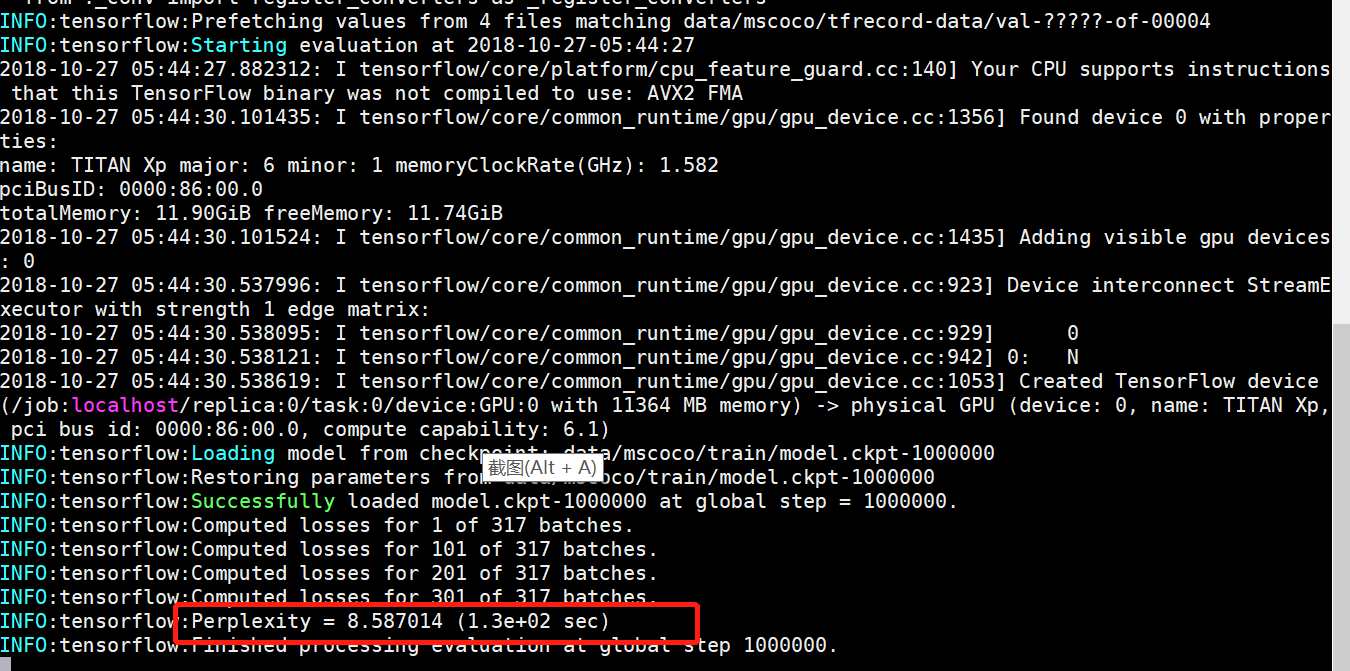
可以看出，在V4的训练下，可以检测出这个是一个小孩，效果略好于V3

## 2.3 Inception-Resnet-v2与Inception-v3的比较

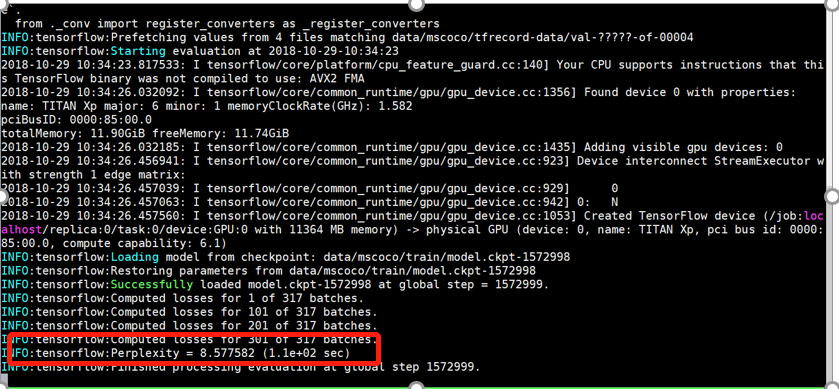
* Inception-v3训练到100w step时，loss在2上下，当时没有截图保留。Inception-Resnet-v2训练到100w step时，截图如下， loss也在2的上下，未见显著提升，可能还需要更多的训练来观察。



* Inception-v3训练到100w step时，运行evaluate得到的困惑度是8.9。没有保留当时的截图和checkpoint文件，但记得这个数值。Inception-Resnet-v2训练到100w step时，运行evaluate得到的困惑度是8.587，这个与inception-v3相比，明显有提升。



* Inception-Resnet-v2项目截至前，最多训练到157w step，运行evaluate得到的困惑度是8.577，既然图如下。较100w step时由提升了0.01，可以看出，还有提升，但提升较困难。



## 2.4 未来得及完成的工作

Encoder部分本想将Inception-v3，inception-v4，inception-resnet-v2三种模型都跑200w step，然后观察loss的情况，困惑度的数值，以及inference的结果。有这样想法的原因是，运行到100w step时，loss的情况差不多，没有大区别，所以想看200w step是否有差别。但每个模型运行200w step大概各自需要6天，因为还要同时调试attention，所以没有运行完。

另外还想将三种模型都试一下在运行100w step后，将inception部分的参数训练打开，一起训练，观察效果如何，但也因时间问题，仅对inception-v4打开训练了inception部分参数，打开训练Inception参数时，loss有增大，还不能确定对最终结果是否有提升。

# 3.Attention layer的引入

基础模型在进行句子中每一个词的预测时，没有考虑到图片中相应的位置，接受的输入仅仅是上一步预测的词和上一步的隐藏层输出，attetion机制就是要在预测每一步的单词时加入这一步单词所对应的图片中位置的信息。

该部分的思路如下：

首先，研究attention模型的结构，理解attention机制是如何工作的。

然后，了解Inception-v4和Inception-Resnet-v2的网络结构，选取特征提取的某个卷积层作为encoder的输出，将其移植到im2txt的image-embedding中。

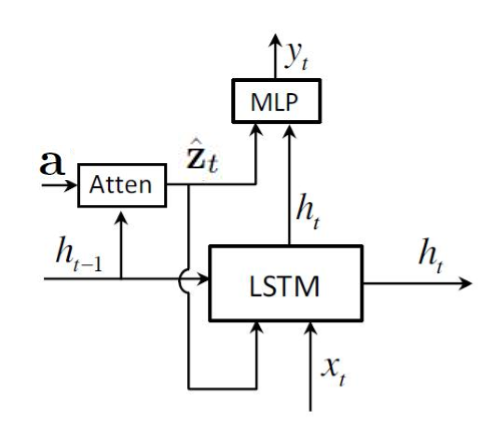
其次，改造train的模型部分，将带有attention的模型训练起来

再次，改造evaluate的运行代码，得到带有attetion模型的困惑度

最后，改造inference的模型和运行代码，得到预测结果。

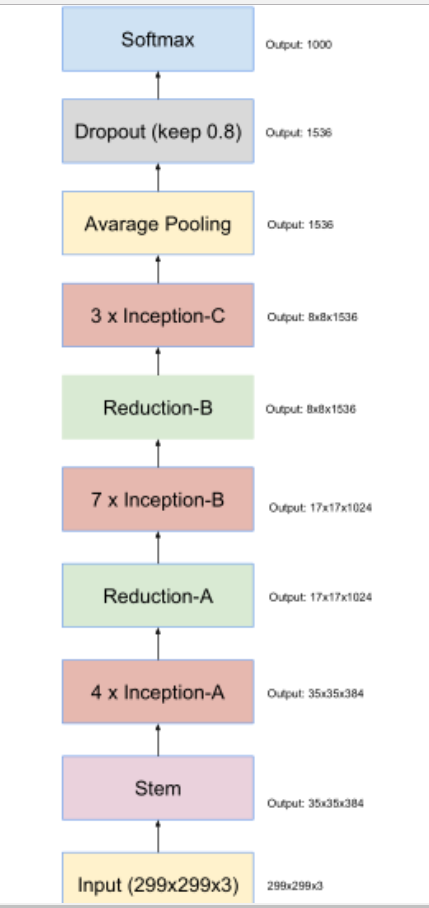
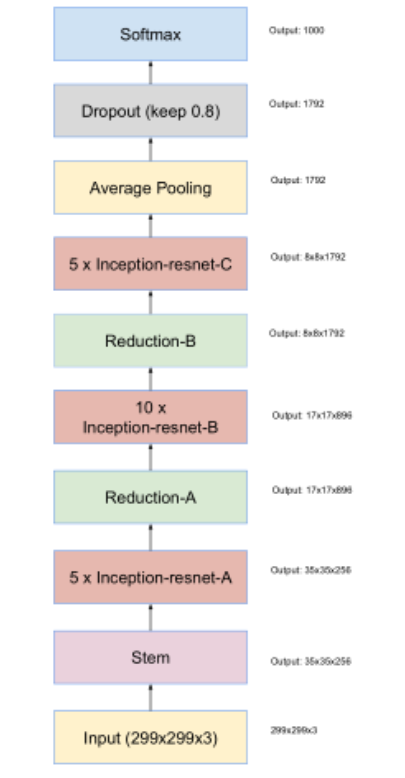
## Attention模型

Attention模型重点在于attention layer怎么插入到基础的模型中，如下图所示。



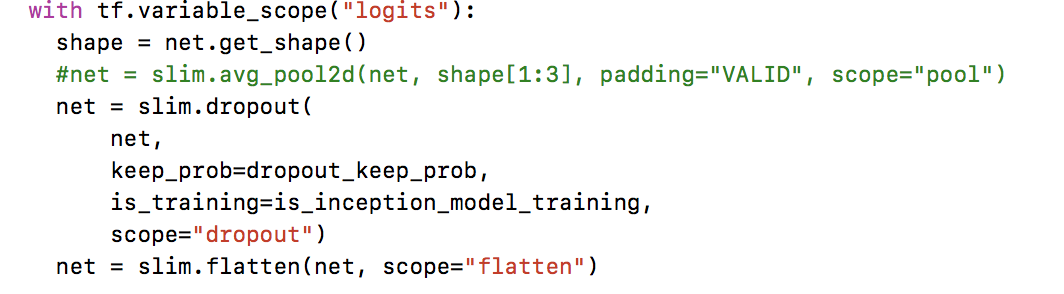
## Inception-v4和Inception-Resnet-v2的网络结构

下图中，左侧是Inception-v4的网络结构，右侧是Inception-v3的网络结构，策略是先都选取最后一个卷积层，后续有机会再看如果使用其他卷积层，对比结果会有什么变化。

选取卷积层时，在如下位置，填写endpoint的名字，就可以取到想要的输出，但是不能对该输出再进行全连接操作，而是直接作为image\_embeddings的输出，用于后续模型输入。

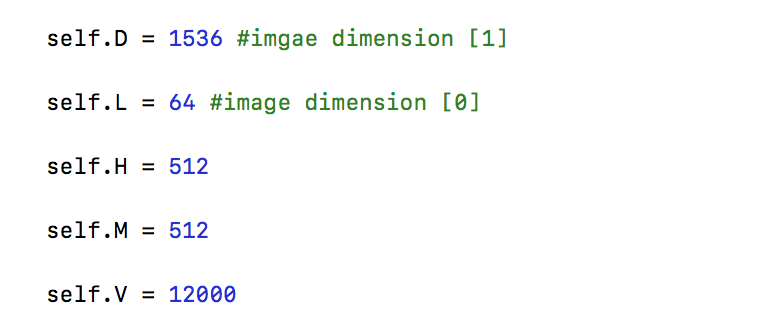
* Image\_embedding.py文件的修改：



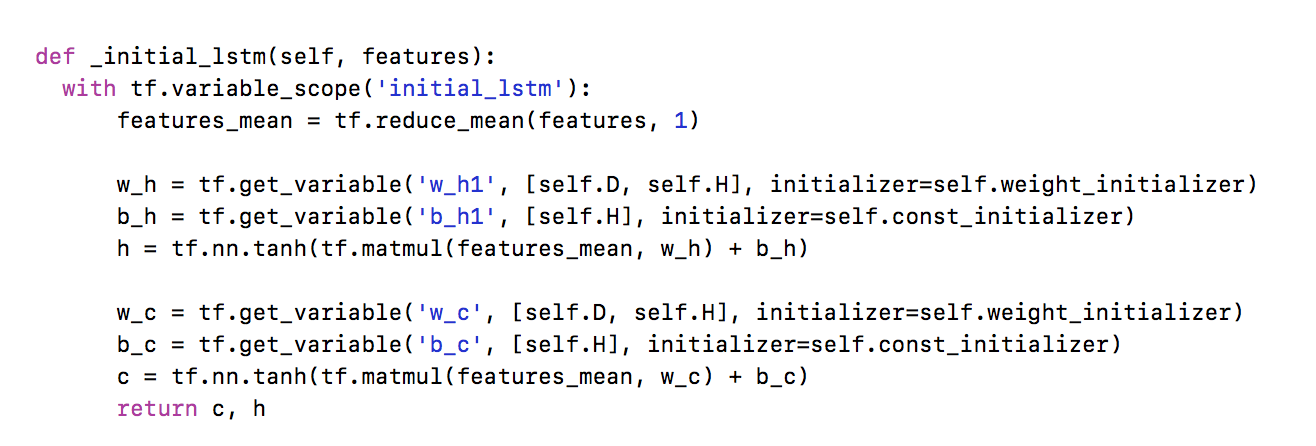
* Show\_and\_tell\_model.py文件的修改：

修改基于Inception-v4作为encoder，舍弃image embedding，去最后一层8\*8\*1536的feature

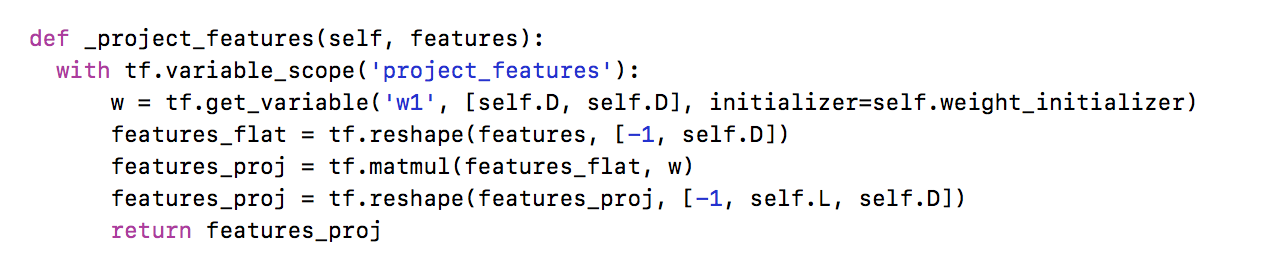
添加的参数设置：

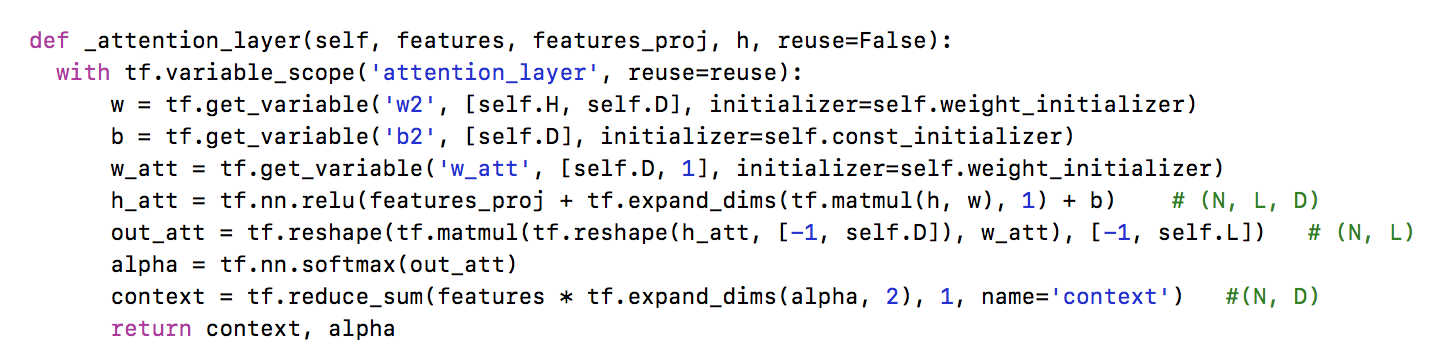


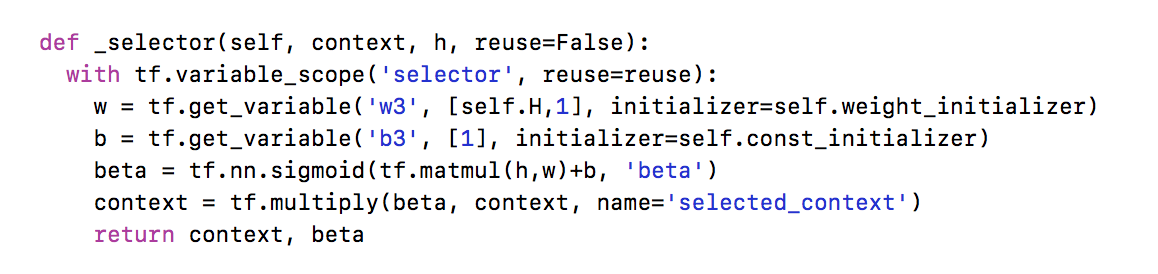
初始化lstm的state：



添加attention layer用到的几个function：



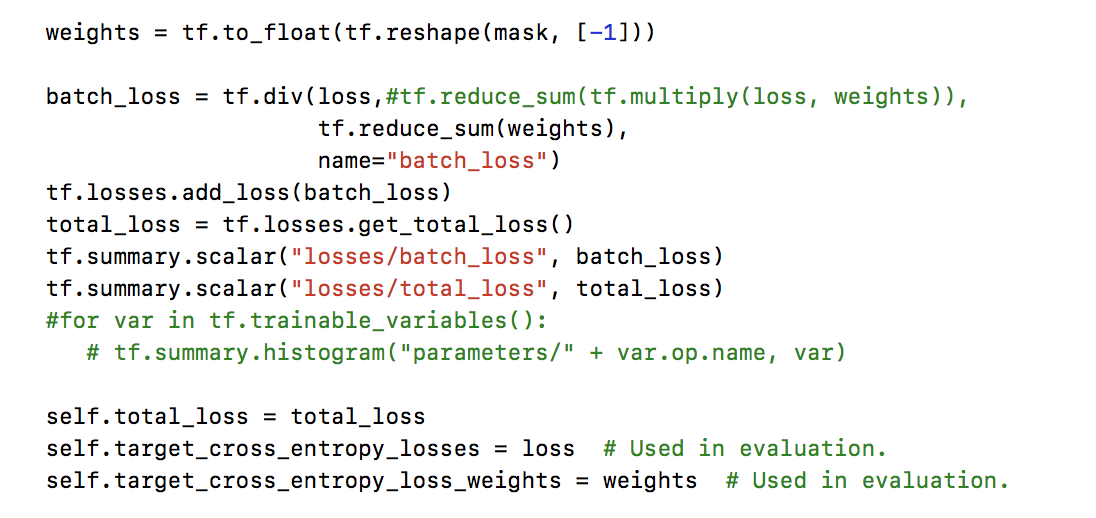






Buildmodel 函数的重建：



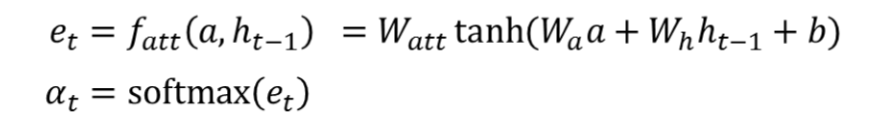


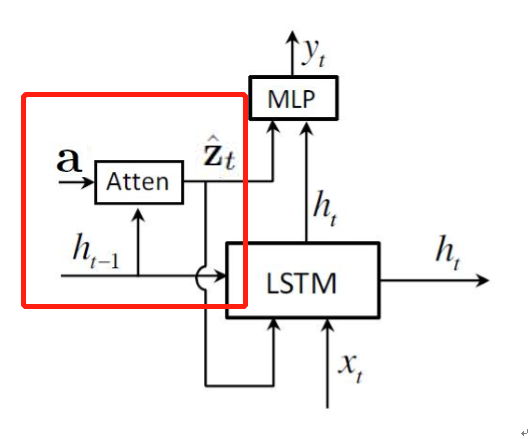
## 在train阶段，增加attention layer，得到合理loss

本节参考了github上<https://github.com/yunjey/show-attend-and-tell>的实现，但在将attention在im2txt上实现，踩了很多坑。包括train，evaluate，inference每个阶段都遇到了不少问题，下面重点说明一些主要的问题和困难。

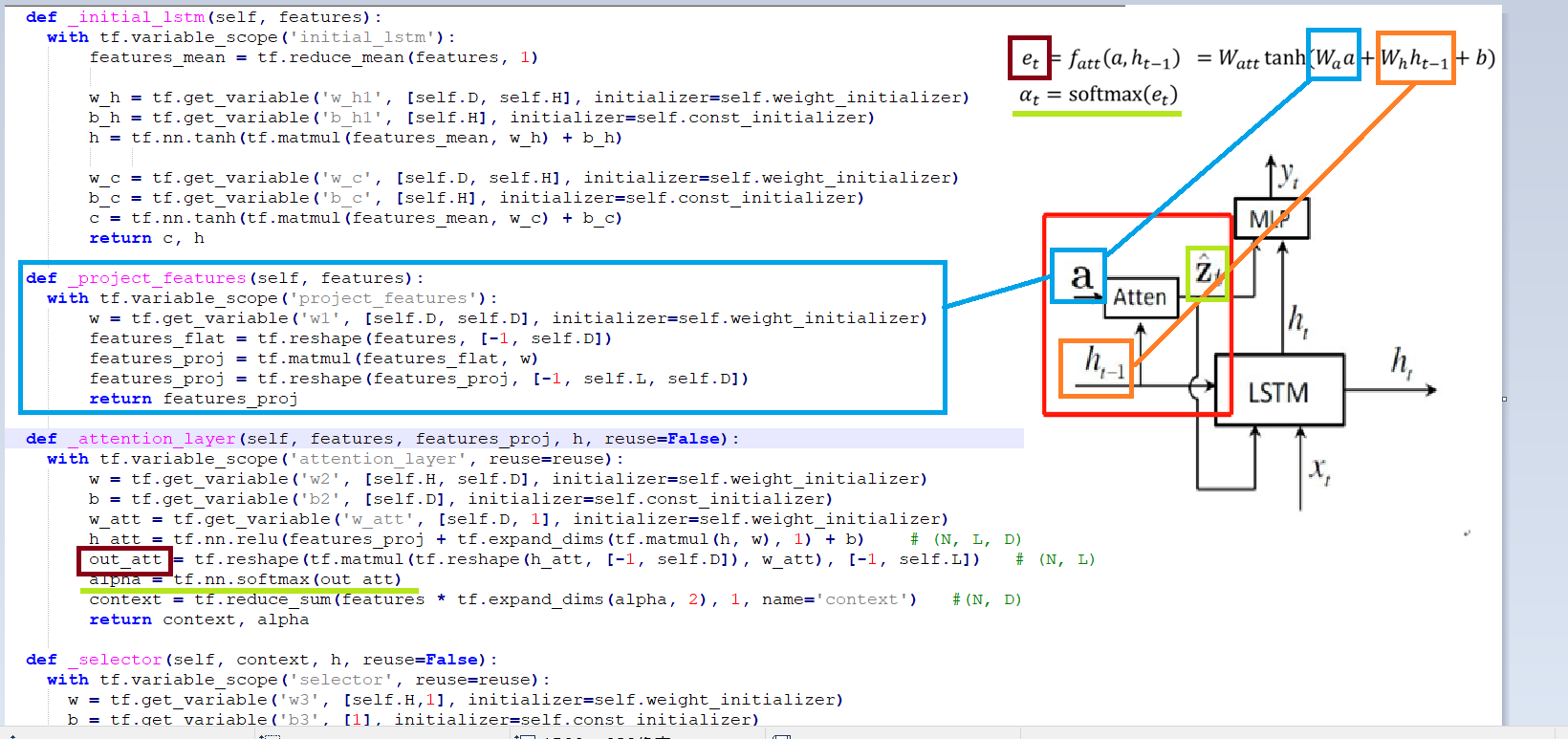
### 3.3.1 attention layer的实现

增加attention layer的代码，attention layer的实现主要是下图中红色框中部分按如下公式进行实现



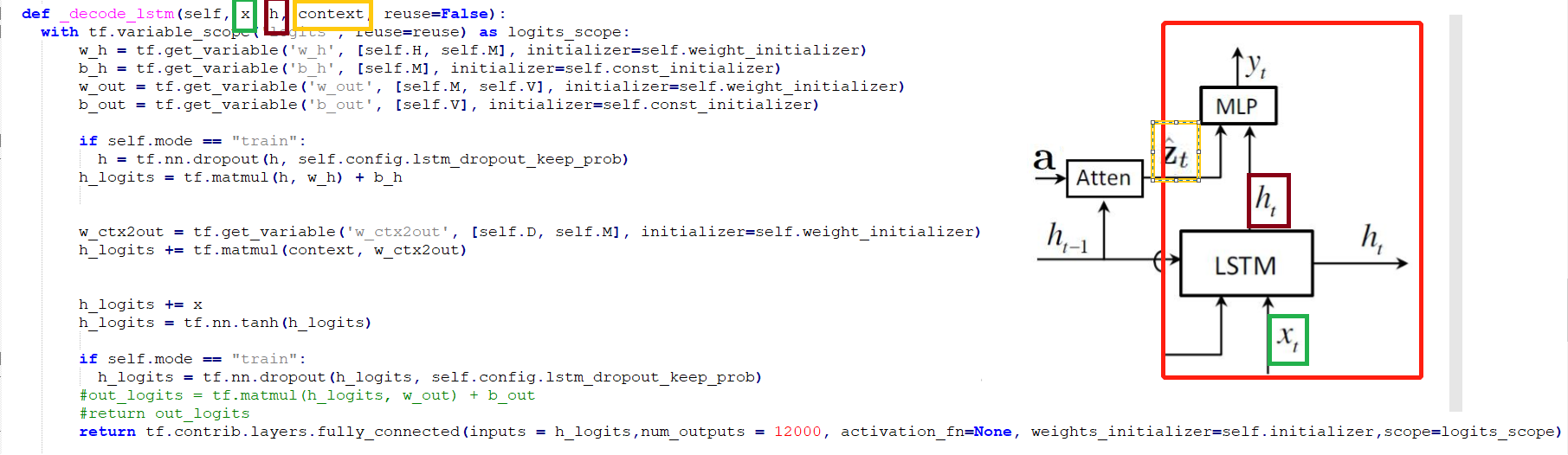


涉及的代码包括如下几个主要函数，下图中尽量将公式，代码以及attention layer中的相应部分用相同颜色标出，方便理解：



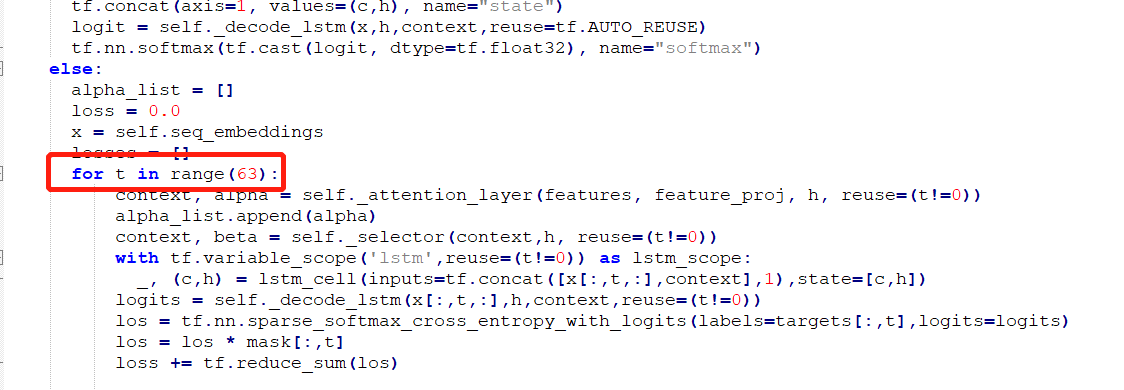
### 3.3.2 decode lstm的实现

加入attention layer后，decoder部分的对应关系如下图所示，尽量用相同颜色给出了对应关系，细节不是特别好标注：



### 3.3.3 按照time step进行循环

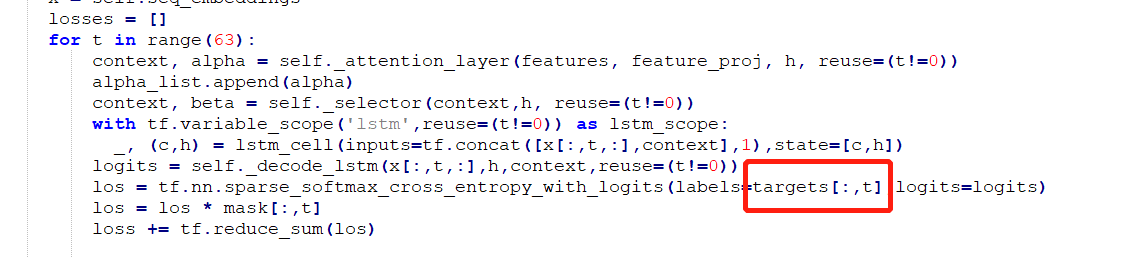
基础模型中，直接使用封装好的dynamic rnn即可，但因为加入了attention layer，只能自己重新写这个循环。循环的代码如下图所示：



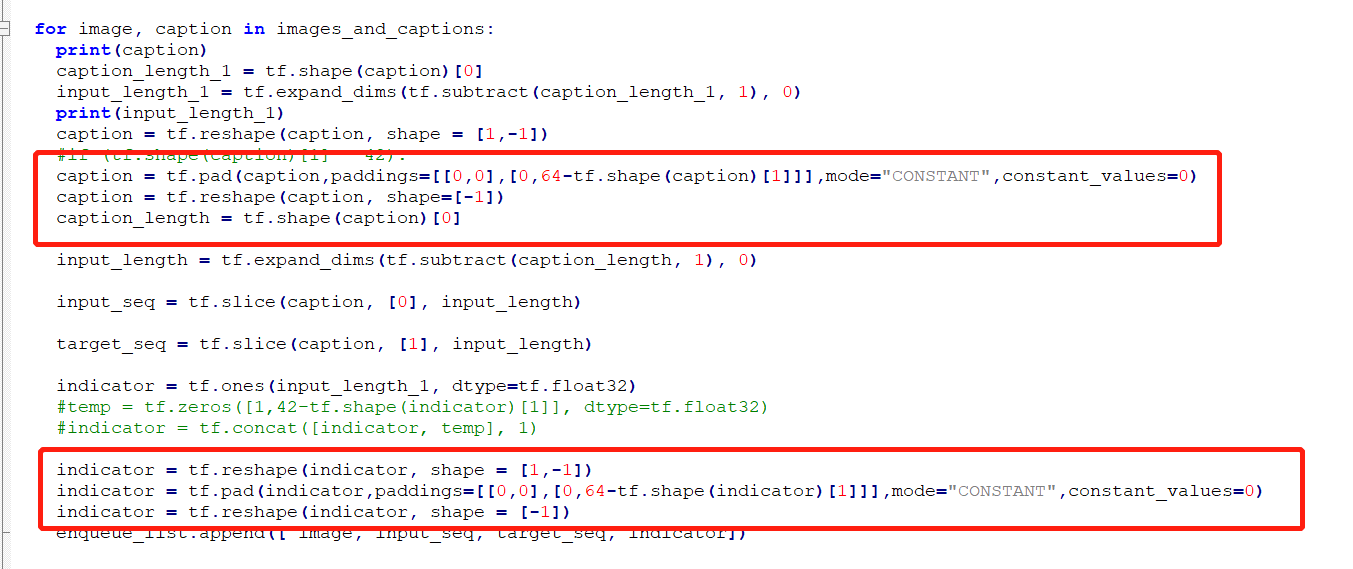
循环时，time step其实就是caption length-1的长度，所以t的范围使用多大比较合适，取决与caption最长是多少。分别试验了16，64，128。发现64已经够用，16不能满足所有的caption长度，所以目前选用了64。

### 3.3.4 对caption进行padding

写好上面的循环后，开始没有对输入的caption及mask进行padding操作，就发现在运行起来后，总是如下的位置越界。



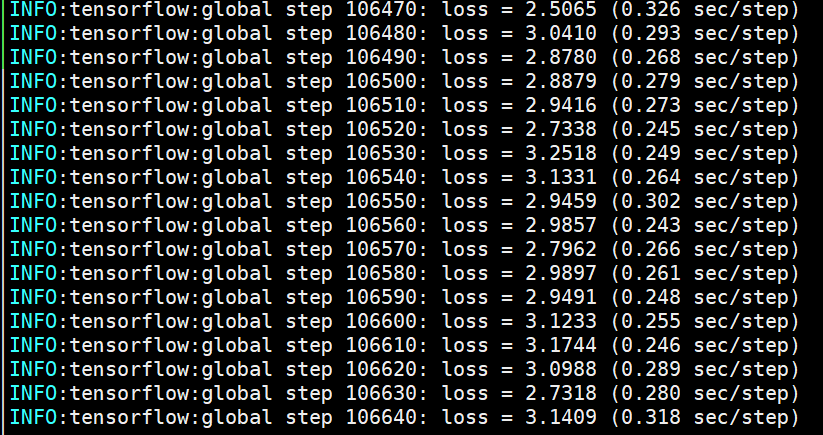
经过我们内部讨论，确认问题所在，因为是重新写for循环，所以需要将caption输入padding到相同长度，这个padding的操作需要我们在input.py中自己完成，所以加上了如下的代码：



增加好padding的代码后，train的过程可以正常运行起来了。

### 3.3.5 loss值偏大，模型可能存在问题

按照上一步修改后，train的过程可以运行了，但是发现loss始终在1000左右震荡，因此对代码继续进行分析，发现是batch loss直接等于的loss，没有除以sum weight，所以loss值很大，修改后loss落在合理区间，如下图所示：



### 3.3.6 调整batch size，避免out of memory的错误

在上述调试过程中，因为引入attention layer，参数量增加很多，原有的batch size=32运行时，会出现OOM的错误，调整为16，发现可以正常运行。

## 在evaluate阶段，修改运行代码，得到困惑度

待补充

## 在inference阶段，修改模型代码，正确进行预测（未完成）

### 3.5.1 基础模型中inference的state维度需要修改

基础im2txt模型中，inference模式下，对state的处理使用了concat和split，相当于将lstm的cell state和hidden state进行了合并处理，而加入attention机制后，c state 和 h state在初始化时就是分开的，因此需要对这里进行拆分修改，否则无法运行inference。

### 3.5.2 image\_feed的赋值始终报错，正在分析调试，暂未解决

下图中，右上红框是运行run\_inference的报错，出错的地方指向右下黄色框的位置，提示image\_feed赋值类型不正确。而真正给image\_feed填值的地方是左上那个绿框的位置，调用左上绿框中函数的地方是左下蓝框的那个位置。此问题暂时还未调通，经过前面的排查，猜测问题，还是因为引入attention后，一些scope发生了变化，可能时scope的使用范围不正确，导致运行时报该错误。

