2018年4月3日

跑数据，但是验证及sleep\_eval不知道怎么写。

2018年4月4日

After 44001 training step(s), loss on training batch is 2.26217.

Tensorflow总览

Variables: 创建，初始化，保存，和恢复

变量的初始化必须在模型的其它操作运行之前先明确地完成。

2018年4月8日星期日

Batch normalization学习笔记

我们把网络中间层在训练过程中，数据分布的改变称之为：“Internal Covariate Shift”。

**2018年4月9日星期一**

今天开始跑第二次的数据，上一次跑数据之所以会准确率会低是因为当时训练集 测试集和验证集的比例设错了，分别是 train data 12789; valid data 14920; test data 14921，上一次跑数据的准确率大概只有75%左右，有点低。这一次重新跑的数据，设置的比例为 train data 29841; valid data 6394; test data 6395。

这个数据收敛的速度这么慢是不是我的learning rate 设置的太低了才0.01?

怎样使用已经保存的模型接着跑，并且继续保存模型。

**2018年4月11日星期三**

怎样把已经读取的tensor中的值直接赋值给新的变量（张量）。

原因还是没有找到，但是发现不同阶段保存的文件，里面的参数竟然完全一模一样，比如layer3-conv2/bias/ExponentialMovingAverage 里面的参数都一样，还有后来断网后跑的数据，模型，里面保存的参数少了 layer3-conv2/bias 这个类型的，原来的类型是保存了两套数据。

通过对mnist保存文件的调查，发现保存数据就应该是保存两套的，

get\_tensor\_by\_name 这个函数是怎么用的呢？？？

tf.get\_default\_graph().get\_tensor\_by\_name(“add:0”)

读了一篇文献《Learning to Compare: Relation Network for Few-Shot Learning》这篇文献中提到了元学习等问题，然后我稍微研究了一下，发现里面很多都需要用到循环神经网络的知识。

* 胶囊网络在阿尔兹海默症的应用
* 元学习 让神经网络学习比较来实现少样本学习
* PCANet
* 通过删除神经元来理解深度学习

整理今天看到的东西：五类的分类准确率达到85.4%。跑数据的文件和检测的文件分别是sleep\_train\_fuwuqi.py 和 sleep\_eval2\_continue1.py。

最近打算开始写文章，现在关于睡眠的思路有两种，一种是走CNN的路子，也就是我现在正在做的。第二种是走RNN的路子感觉这个已经有人在做了。不管哪一种都要好好的研读文章。

**2018年4月13日星期五**

画混淆矩阵。

在之前保存的数据中，0标签的也就是清醒期的大概有259700个epoch，总37个被试。

所用到的通道为sleep\_EEGPzOz，有35630个epoch。

tf.argmax就是返回最大的那个数值所在的下标。

**2018年4月14日星期六**

对标签为0的数据来说，对应的就是清醒期，{0: 10339, 1: 2, 2: 107, 3: 55, 4: 4}

对标签为1的数据来说，对应的就是N1期，{0: 2175, 1: 55, 2: 374, 3: 51, 4: 9}；感觉数据的准确率有点低。

对标签为2的数据来说，对应的就是N2期，{0: 3416, 1: 5, 2: 10519, 3: 2722, 4: 14}

对标签为3的数据来说，对应的就是N3期， {0: 715, 1: 0, 2: 106, 3: 4679, 4: 0}

对标签为4的数据来说，对应的就是R期， {0: 4975, 1: 5, 2: 1825, 3: 342, 4: 136} 只有60%左右的准确率，这显然不对。

要想数据不拟合，至少需要26000张图片。考虑到数据的均衡性，共五类，每类至少5200张照片。

**2018年4月14日星期日**

今天跑数据第一次错误，发现在进行train\_test\_split 函数随机分割数据的时候，如果random\_state 不填或者为0，那么每次产生的随机数都不一样，如果随机种子不为0，则每次产生的数据都是一样的。

W0：accuracy\_score：0.9887694

N1： accuracy\_score：0.049174175 results\_1

N2： accuracy\_score: 0.5269849

N3: accuracy\_score: 0.9129091

R： accuracy\_score: 0.043800633

Sleep\_experiment2 PzOz文件夹任务跑完了，准确率为0.849332% loss: 0.49175

sleep\_experiment3\_FpzCz 文件夹，准确率为0.822305% loss: 0.743832

Sleep\_experiment4\_zong 文件夹，准确率为0.832777% loss: 0.144453

今天主要跑的数据主要有三个类型，两个单通道和一个双通道的数据。

2018年04月17日

今天上午，文件夹的名字为 sleep\_experiment2\_same，跑的任务是把sleep\_inference.py文件padding的值从valid 改成了 same。

Sleep\_experiment2 PzOz文件夹任务跑完了，准确率为0.85803% loss: 0.898171

sleep\_experiment2\_same 文件夹任务跑完了，准确率为0.861292% loss: 0.848106

今日任务：

尝试一下用保存的文件，再次重新跑数据。 文件夹的名字为 sleep\_experiment2\_restore

每次跑数据之前，一定要注意保存的路径。

胶囊网络的实现CapsNet-Tensorflow

求F1-score recall precision 等等是只用测试集（只有20%的吗）还是用所有的？？？看文献？？？

路由是指[路由器](https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1%E5%99%A8" \t "https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1/_blank)从一个接口上收到[数据包](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%8C%85" \t "https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1/_blank)，根据数据包的目的[地址](https://baike.baidu.com/item/%E5%9C%B0%E5%9D%80" \t "https://baike.baidu.com/item/%E8%B7%AF%E7%94%B1/_blank)进行定向并转发到另一个接口的过程。

['calculate\_test\_sleep.py', 'image\_0', 'labels\_0']

accuracy\_score::::: 0.40973783

results::::: 0.40973783

['calculate\_test\_sleep.py', 'image\_1', 'labels\_1']

accuracy\_score::::: 0.056169428

results::::: 0.056169428

['calculate\_test\_sleep.py', 'image\_2', 'labels\_2']

accuracy\_score::::: 0.34671533

results::::: 0.34671533

['calculate\_test\_sleep.py', 'image\_3', 'labels\_3']

accuracy\_score::::: 0.94530046

results::::: 0.94530046

['calculate\_test\_sleep.py', 'image\_4', 'labels\_4']

accuracy\_score::::: 0.030392883

results::::: 0.030392883

image\_0 {0: 570, 1: 70, 2: 275, 3: 391, 4: 29}

image\_1 {0: 532, 1: 61, 2: 197, 3: 276, 4: 20}

Image\_2 {0: 269, 1: 20, 2: 499, 3: 574, 4: 8}

Image\_3 {0: 62, 1: 1, 2: 12, 3: 1223, 4: 0}

Iamge\_4 {0: 593, 1: 31, 2: 354, 3: 330, 4: 41}

正确 2394 总的 6438 只有37%的准确率

再次进行测试。用的是sleep\_eval中的数据来直接进行计算的。

Image\_0: 1335; Image\_1: 1086; Image\_2: 1370; Image\_3: 1298; Image\_4: 1349;

array([[1302, 21, 1, 0, 11],

[ 98, 649, 117, 9, 213],

[ 7, 24, 1261, 41, 37],

[ 0, 0, 14, 1284, 0],

[ 21, 159, 93, 4, 1072]])

In [36]: f1score = f1\_score(y\_true, y\_pred, average='micro')

In [37]: f1score

Out[37]: 0.8648648648648649

####求混淆矩阵 f1-score 等等

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import precision\_score, recall\_score, f1\_score

results = sess.run(tf.argmax(y, 1), feed\_dict={x: reshaped\_xs, y\_: ys})

y\_pred = results

y\_true = np.argmax(ys, 1)

confusionmatrix = confusion\_matrix(y\_true, y\_pred) ####直接求混淆矩阵

f1score = f1\_score(y\_true, y\_pred, average='micro'

2018年04月18日

任务：把总的数据，改成same再跑一下，记得用已经保存好的模型参数。

**2018年4月22日星期日**

CSDN密码 LinBin\_85623232

把文件改一下，还是直接写文献。

这个任务已经在run了，接下来就是看文献。写文章。 完成文献 Data Acquisition部分的写作。

《统计学习方法》第六章的学习，并且认真做好笔记，这本书，最近这一周学会吧，里面好多知识我都会了。

《python计算机视觉编程》学习，这是一个比较实用的课程，最近一段时间学会吧（两周）。

2018年4月23日星期一

刚刚测试完 sleep\_experiment4\_zong\_same 两通道的数据，padding都为same， 然后测试出来的结果为 83.54% loss为 0.23吧

主要跑的数据要以下几个：

* 单通道PzOz FpzCz 通道的 padding为 SAME 和VALID 共两层隐藏层的网络
* 双通道融合的padding为 SAME 和VALID 共三层隐藏层的网络
* 还需要再跑的是 双通道融合的padding为 SAME的 共两层隐藏层的网络

今天的任务：晚上再跑一个 双通道融合的padding为 SAME的 共两层隐藏层的网络

###############################################################################

**未曾见过一个早起、勤奋、谨慎、诚实的人抱怨命运的不好；良好的品格，优良的习惯，坚强的意志，是不会被假设所谓的命运击败的。**

2018年04月15日星期日

今天上午跑步加健身，学到了一句话，长得好看的人都喜欢健身。感觉自己学习不是很用功，不够努力。

今天看到一篇雷军的软文，里面讲到，如果你不是富贵人家出生，今生你在职场成就一番事业的唯一机会是选择忠诚和勤奋。