# 实验记录

## 1时间：20190602，13：00-？

## 2实验目的：

在探究使用0.3cut off数据集是否会得到类似结果。

## 3实验代码存放文件夹：

20190602中。

## 4实验运行时间：

1天及以上。

## 5实验使用数据信息：

药物作用位点预测，可能存在的药物作用位点标志为1，正样本。其余其他位点均为负样本。消除位于边界的负样本，包含正样本位点的负样本。

## 6.1，数据集正负样本设置，比例设置，

正负样本1：33。采用十则交叉验证模式构建10个数据集的数据，依次进行学习。目前没完成整体评估+存取模型+选取最优模型等步骤

## 6.2，网络设置：

def zyh\_CNN(trainX, trainY,valX=None, valY=None, compiletimes = 0, forkinas=False,transferlayer=1,compilemodels=None,earlystop=None,nb\_epoch=【5】,batch\_size = 8192,fildername = None ):

"""

:argument

:return:

"""

input\_row = trainX.shape[2]

input\_col = trainX.shape[3]

trainX\_t = trainX;

valX\_t = valX;

#print(input\_row)

#print(input\_col)

trainX\_t.shape = (trainX\_t.shape[0], input\_row, input\_col)

if (valX is not None):

valX\_t.shape = (valX\_t.shape[0], input\_row, input\_col)

#print(valX\_t)

if (earlystop is not None):

early\_stopping = EarlyStopping(monitor='val\_loss', patience=earlystop)

nb\_epoch = 1000; # set to a very big value since earlystop used

if compiletimes == 0:

input = Input(shape=(input\_row, input\_col))

filter1 = 64

filtersize1 = 2

dropout1 = 0.25

L1CNN = 0

nb\_classes = 2

batch\_size = batch\_size

actfun = "relu";

nadam = Nadam(lr=0.00001)

optimization = nadam

dense\_size1 = 128

dense\_size2 = 64

dense\_size3 = 8

dropout\_dense1 = 0.298224

dropout\_dense2 = 0

dropout\_dense3 = 0

input = Input(shape=(input\_row, input\_col))

x = conv.Conv1D(filter1, filtersize1, init='glorot\_normal', W\_regularizer=regularizers.l2(L1CNN),

border\_mode="same")(input)

x = Dropout(dropout1)(x)

x = Activation(actfun)(x)

x = core.Flatten()(x)

output = x

output = Dropout(dropout1)(output)

output = Dense(dense\_size1, init='glorot\_normal', activation='relu')(output)

output = Dropout(dropout\_dense1)(output)

output = Dense(dense\_size2, activation="relu", init='glorot\_normal')(output)

output = Dropout(dropout\_dense2)(output)

output = Dense(dense\_size3, activation="relu", init='glorot\_normal')(output)

output = Dropout(dropout\_dense3)(output)

out = Dense(nb\_classes, init='glorot\_normal', activation='softmax')(output)

cnn = Model(input, out)

cnn.compile(loss='binary\_crossentropy', optimizer=optimization, metrics=['accuracy'])

pass

else:

print("use old net")

cnn = compilemodels

pass‘

之后使用输出模式查看网络。

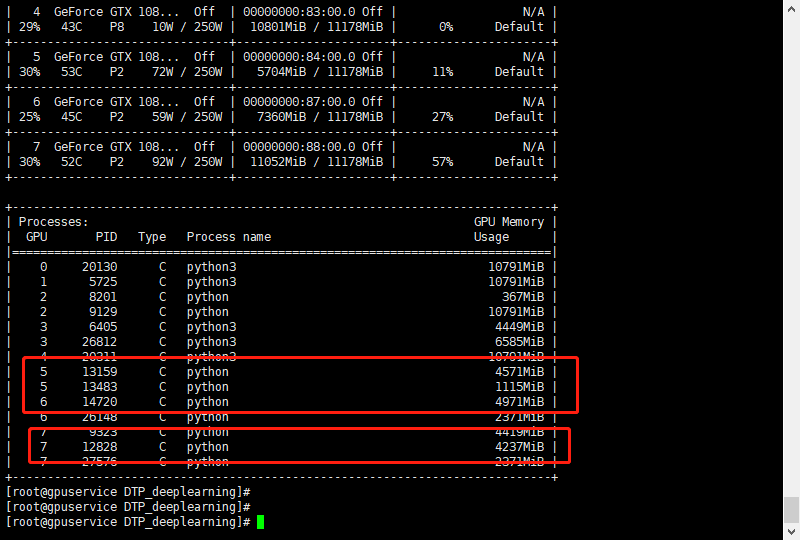
## 7实验变量：

网络设置中的epoch：分别设置为3\6\9,batch设置为8192、4096.

## 8实验运行记录

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 开始时间 | 输出文件 | 结束时间 | GPU编号（数量） | 运行id | Epoch | Batch size |
| 1 | 中午13：00 | 20190602\_01.ou |  | 7 | 12828 | 3 | 8192 |
| 2 | 中午13：00 | 20190602\_02.ou |  | 5 | 13159 | 6 | 4096 |
| 3 | 中午13：00 | 20190602\_03.ou |  | 5 | 13483 | 9 | 4096 |
| 4 | 下午 |  |  | ----- |  |  |  |

存放文件夹：1、2、3



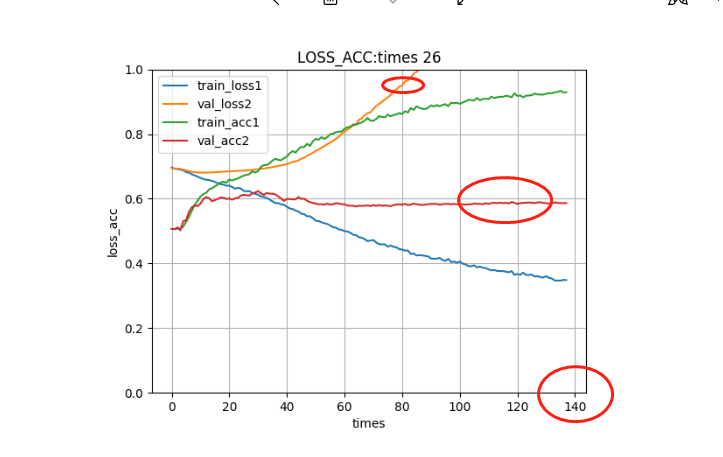
第13483号进程（第三组实验）为什么占的显存那么小？

## 9实验结果存放位置

将图片下载后存储

## 10结果分析

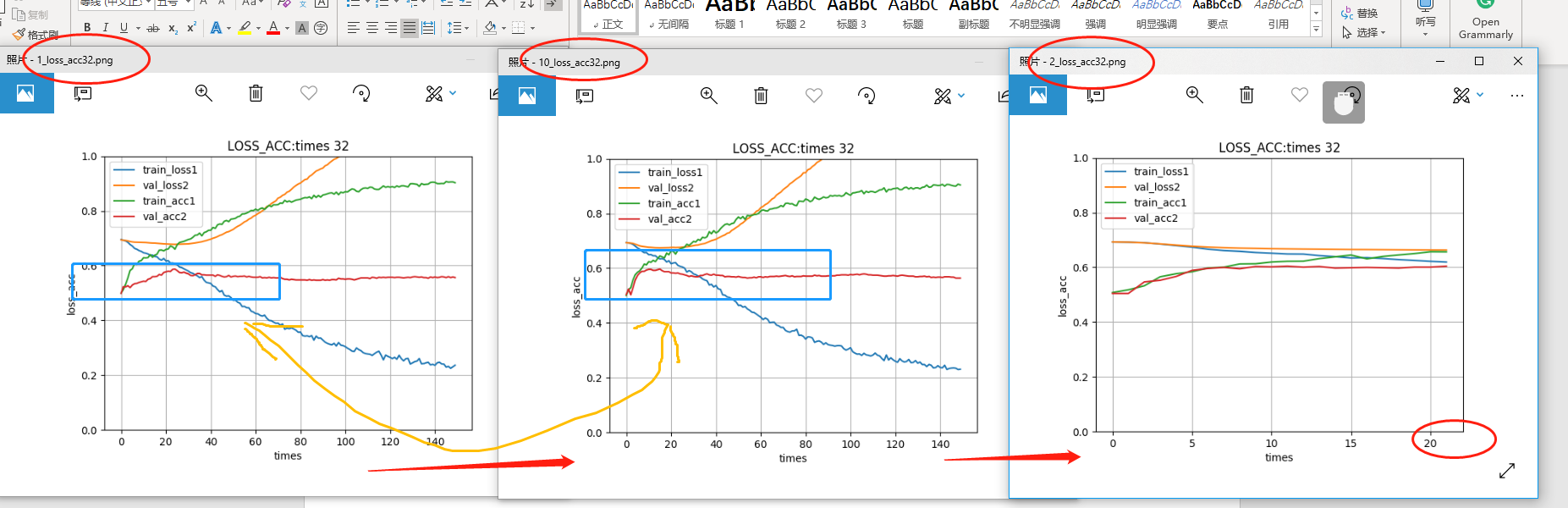
【3号实验结果】



这个是0.9去冗余后数据，epoch =9训练得到的数据结果。在第一折中迭代输入全部数据140次。

可以看到前10次迭代还是学到一点东西，之后loss飞起，val基本保持0.6.。。。loss飞起表面，没有学到东西。（其余曲线不看了）

【2号实验结果】



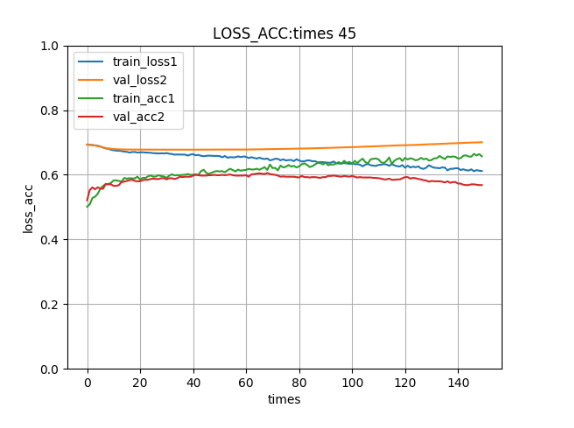
2号实验运行时间较长，所以得到的数据组数多

Batchsize-和epoch的次数分别改变（我可能当时比较迷糊，竟然没有设置控制变量。。。导致不知道怎么看到对比）

根据组间我们发现loss飞出边界的训练轮次在80-100之间，不知有无可解释性。（1号实验中也是，但是epoch不同，batch不同，难道它俩抵消了？）

蓝色框中，学习到的模式是不同的，我们也能看出其中先后放入数据不同产生的变化。

【1号实验】



Epoch = 3，感觉还没怎么训练，就结束了。。还没有产生巨大的区分，我认为是因为训练量不够。（也可能有粗心设置错参数的可能）

## 11总结

总之，当我们的模型遇到非冗余数据后，训练效果极度下降，则我们需要更换网络、或者更换网络结构。

这就是当前需要做的事情~

