# 2019年1月17日星期四

# 实验记录

## 1时间：19：00-21：00

## 2实验目的：

在泳冰的提议下，设置更大的epoch，查看是否还可以提升效果。

## 3实验代码存放文件夹：

20190116

## 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：

if compiletimes == 0:

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

filter1 = 64（这里采用前一次试验最高数值）

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'])

之后使用输出模式查看网络。

## 7实验变量：

网络设置中的filter1：分别设置为16，32，48，64。

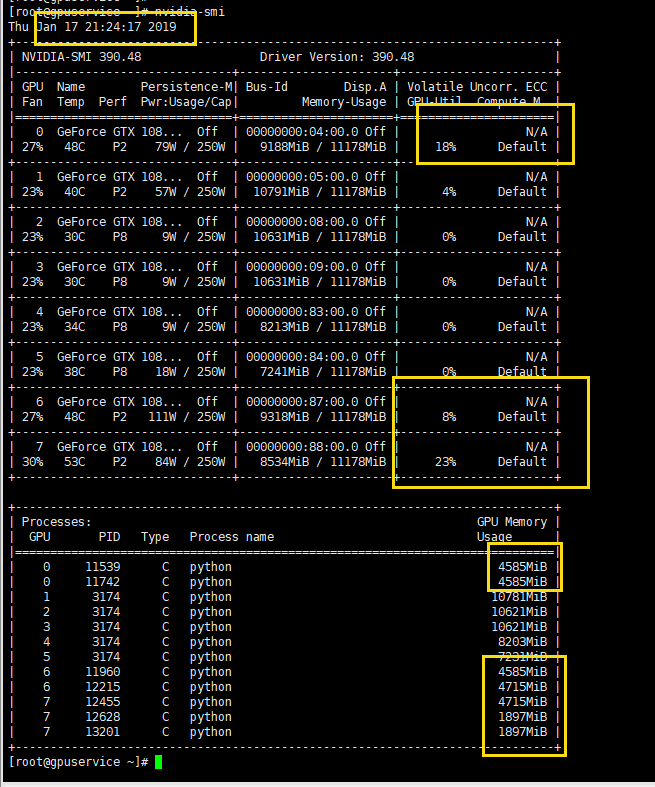
## 8实验运行记录

## ，，，，。

Batchsize = 4096

输出文件名字：test\_实验编号?\_GPU(?)\_日期\_时间.out

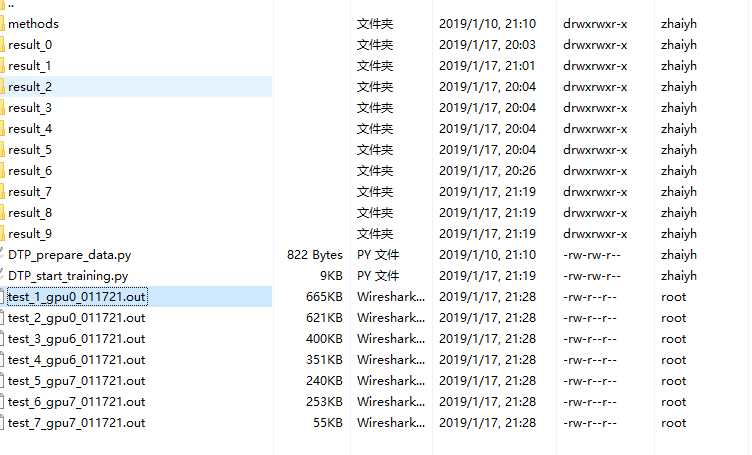
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 输出文件夹 | 输出文件名字 | GPU编号（数量） | 运行id | filters | Epoch | filtersize |  |
| 1 | Result\_1 | 1\_0\_0117 | 0 | 11539 | 64 | 750 | 2 |  |
| 2 | Result\_2 | 2\_0\_0117 | 0 | 11742 | 64 | 1000 | 2 |  |
| 3 | Result\_3 | 3\_6\_0117 | 6 | 11960 | 64 | 1250 | 2 |  |
| 4 | Result\_4 | 4\_6\_0117 | 6 | 12215 | 80 | 750 | 2 |  |
| 5 | Result\_5 | 5\_7\_0117 | 7 | 12455 | 96 | 750 | 2 |  |
| 6 | Result\_6 | 6\_7\_0117 | 7 | 12628 | 64 | 750 | 3 |  |
| 7 | Result\_7 | 7\_7\_0117 | 7 | 13201 | 64 | 750 | 4 |  |



## 针对此图的分析是：

Filters 的改变会影响GPU占用内存。Epoch不会改变内存使用量。Filter增大会降低内存内存使用量。。。

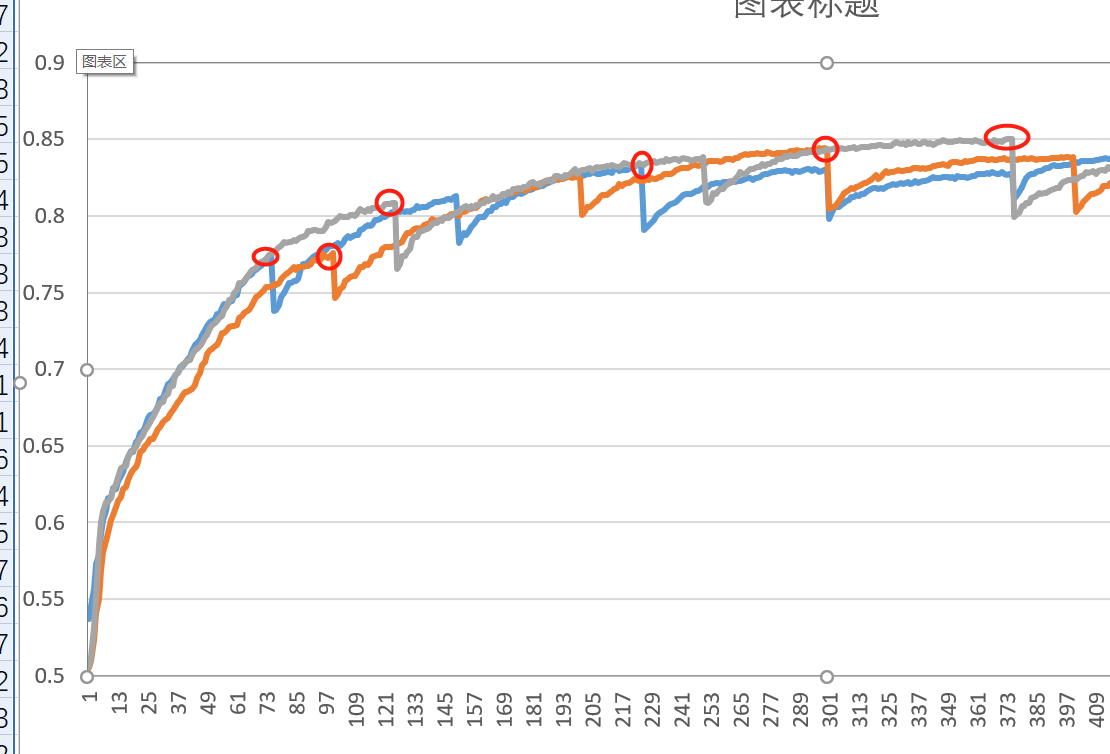
存储文件截图



这是30min内结果，还没有图画出来。明早再看咯！

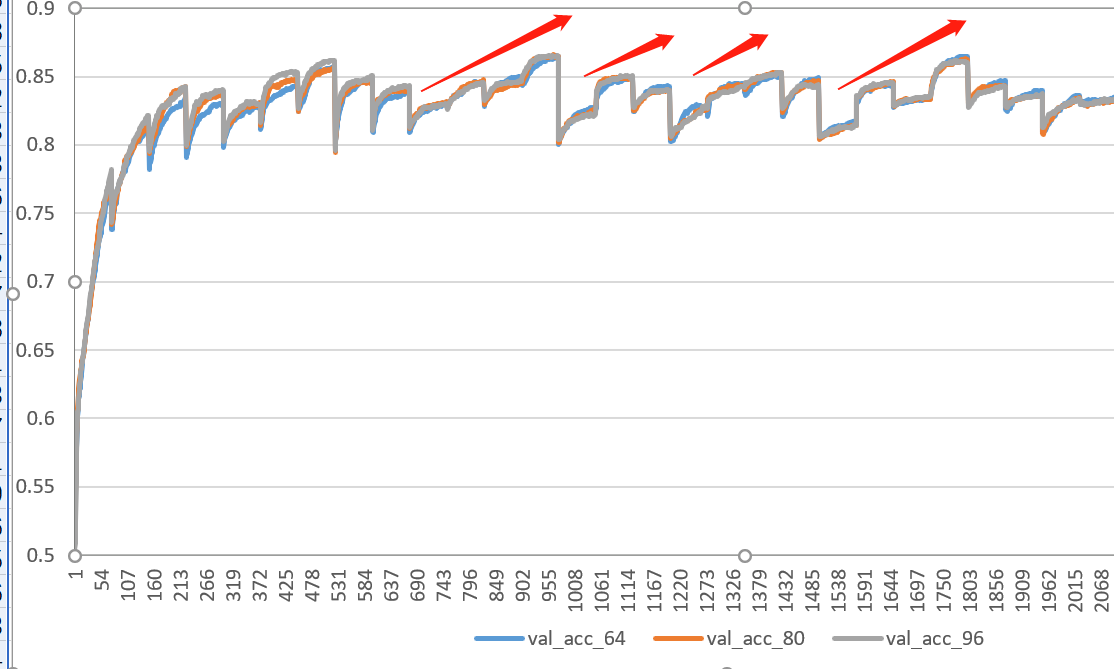
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 输出文件夹 | 输出文件名字 | GPU编号（数量） | 运行id | filters | Epoch | filtersize |  |
| 1 | Result\_1 | 1\_0\_0117 | 0 | 11539 | 64 | 750 | 2 |  |
| 2 | Result\_2 | 2\_0\_0117 | 0 | 11742 | 64 | 1000 | 2 |  |
| 3 | Result\_3 | 3\_6\_0117 | 6 | 11960 | 64 | 1250 | 2 |  |
| 4 | Result\_4 | 4\_6\_0117 | 6 | 12215 | 80 | 750 | 2 |  |
| 5 | Result\_5 | 5\_7\_0117 | 7 | 12455 | 96 | 750 | 2 |  |
| 6 | Result\_6 | 6\_7\_0117 | 7 | 12628 | 64 | 750 | 3 |  |
| 7 | Result\_7 | 7\_7\_0117 | 7 | 13201 | 64 | 750 | 4 |  |

1-2-3第一组结果比较epoch,希望看到更高数值的epoch出现更优秀的结果。

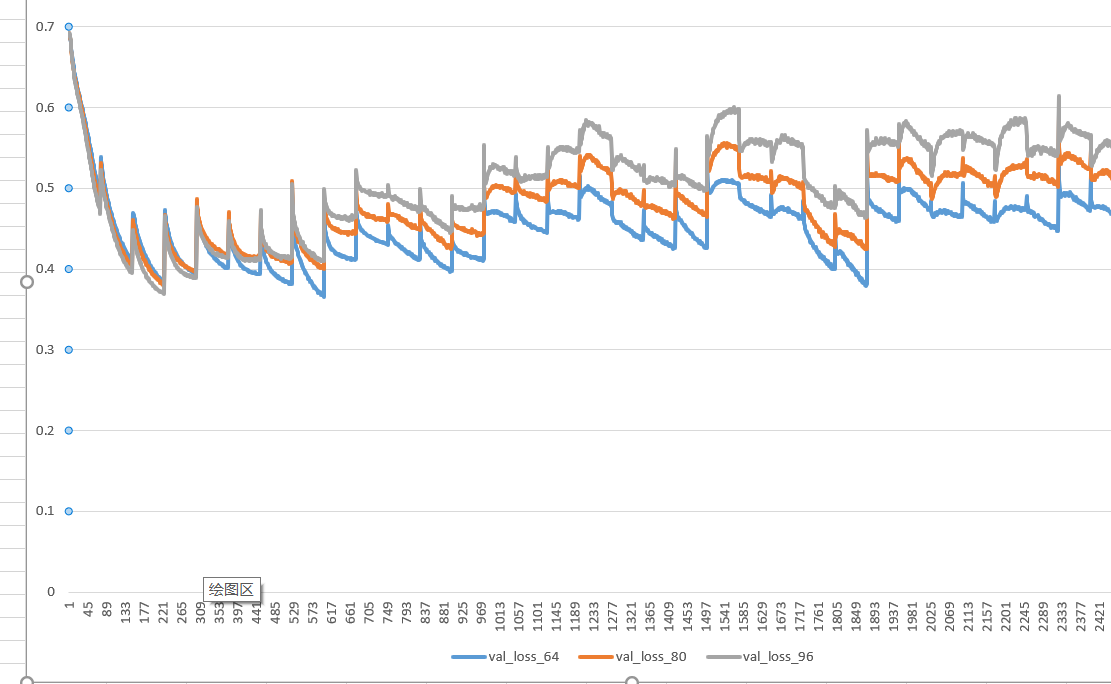


在val\_acc的角度中，我们可以看到，epoch增大后，确实出现了更好的效果：在第一次训练结果的表现中，灰色线数值较高。所以之后设置epoch建议设置在1250左右。

1-4-5比较filters。Filters ,在我看来是第二层神经元的个数。我分别设置了64-80-96.（使用test1中的其他参数）

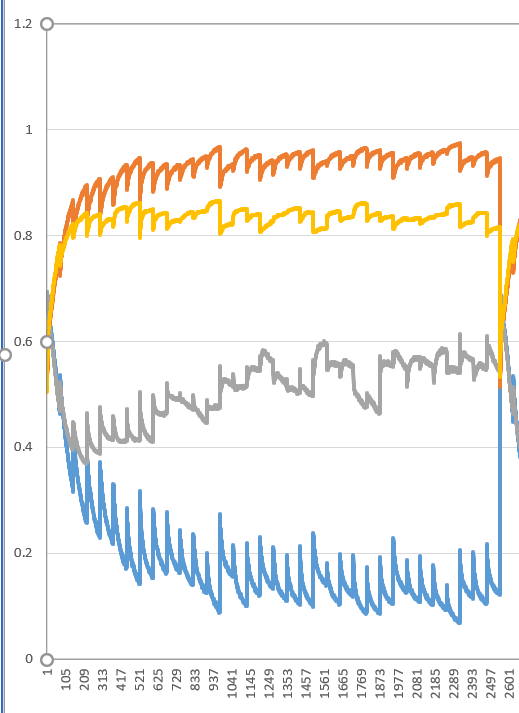


根据前几批次看，发现了96的看起来更好。Val\_acc数据。



而根据val\_loss曲线，我们可以看出，前几批次的训练是好的，后面出现了波动和不规律现象，根据刘喆的建议，是过拟合现象。

对应起来看是这样的。



## 9实验结果存放位置

将图片下载后存储

## 10结果分析

## 11总结