# 实验记录

## 1时间：20190605，23：00-？

## 2实验目的：

随机选取网络中的几个参数进行修改和调整，根据结果查看网络调整对实验结果的影响。

## 3实验代码存放文件夹：

20190605文件夹中。

## 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

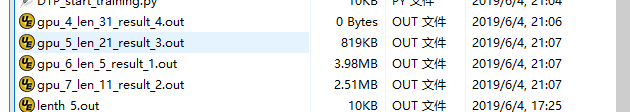
else:

print("use old net")

cnn = compilemodels

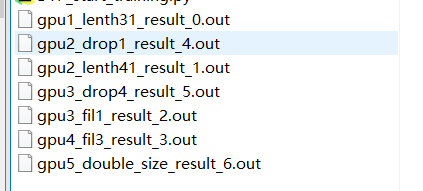
pass7实验变量：

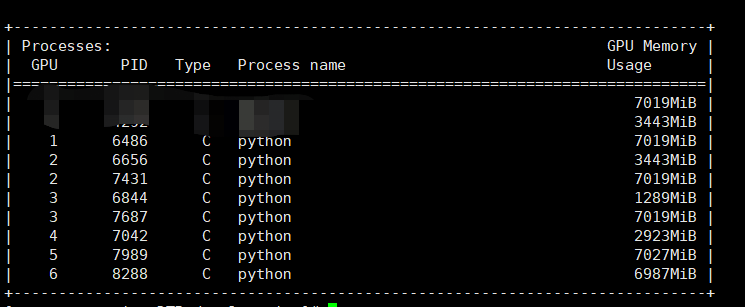
## 8实验运行记录



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 开始时间 | 输出文件 | 结束时间 | GPU编号（数量） | 运行id | 长度 | 文件夹 | Filtersize(kernalsize)  卷积核改变 | Densize改变 | 卷积层drop out改变 | 描述 |
| 1 | 23：00 | 如图 |  | 1 | 6486 | 31 | 0 | 2 | 128：64：8 | 0.25 |  |
| 2 | 23：00 | 如图 |  | 2 | 6656 | **41** | 1 | 2 |  |  |  |
| 3 | 23：00 | 如图 |  | 3 | 6844 | 31 | 2 | **1** |  |  |  |
| 4 | 23：00 | 如图 |  | 4 | 7042 | 31 | 3 | **3** |  |  |  |
| 5 | 23：00 | 如图 |  | 2 | 7431 | 31 | 4 | 2 |  | 0.1 |  |
| 6 | 23：00 | 如图 |  | 2 | 7687 | 31 | 5 | 2 |  | 0.4 |  |
| 7 | 23：00 | 如图 |  |  | 7989 | 31 | 6 | 2 | 256：124：16  （数量加倍） |  |  |
| 8 | 23：00 | 如图 |  |  | 8828 | 31 | 7 | 2 | 128：128：64：64：8：8（层数加倍） |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

以后根据预期主要变量由小到大设置数值，GPU编号，名字，文件夹都要由小到大设置。





## 9实验结果存放位置

## 10结果分析

## 11总结