# 实验记录

## 1时间：20190604，13：00-？

## 2实验目的：

使用新提取的数据，（同文件夹有说明）进行实验，检测新数据数目、存储是否出现问题，针对这样去冗余后的数据，得到初步结果，针对初步结果尝试修改本网络。我相信CNN还是好用的。

## 3实验代码存放文件夹：

20190604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

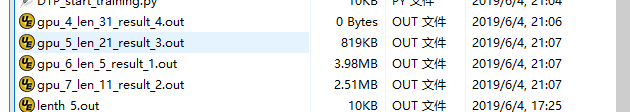
else:

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 |
| 1 | 17：00 | 如图 |  | 7 | 27699 | 11 | 2 |
| 2 | 17：00 | 如图 |  | 6 | 28716 | 5 | 1 |
| 3 | 17：00 | 如图 |  | 5 | 32185 | 21 | 3 |
| 4 | 21：00 |  |  | 4----- | 5709 | 31 | 4 |

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