首先，需要了解CNN的基本概念，这部分的内容可以通过如下网站来进行学习。

**资源**

一、CNN学习

可以通过在吴恩达的Deep Learning上来学习。

<https://study.163.com/course/introduction/1004570029.htm>

接下来可以在优达学城上学习并进行代码的实际操作。

<https://classroom.udacity.com/courses/ud730>

这两个课程都是英文的，但是配有中英双语字幕，而且是2D CNN的，大部分的实验都是基于单张图片形式的，介绍两个在编写代码时可供使用的数据集MNIST和notMNIST。

关于3D CNN的学习可以参考这篇博客<https://www.cnblogs.com/Ponys/p/3450177.html>

，因为3D CNN目前大多的应用都是在视频中人行为的识别上，因为只有连续且具有同一动作属性的视频帧才可以被连在一起并被赋予相同的标签，这是3D-CNN-Keras关于视频帧中人行为的识别代码：

<https://github.com/Ectsang/3D-CNN-Keras/blob/master/3d-cnn-action-recog-keras.py>

二、数据获取

数据获取可以见我之前写的博客：

<https://blog.csdn.net/qq_31550269/article/details/86543717>

不过如果对所需要的股票数据变量要求不高（例如，只需要开盘，收盘，最高，最低，交易量）的话，可以从Yahoo！Finance上

<https://finance.yahoo.com>上搜索单支股票并进行下载，一开始训练模型时可以用少量的行情指标变量，然后再一点点增加变量的个数。

其中，在<https://cn.investing.com/indices/nyse-market-composite-components>上可以搜索到不同市场内所包含的成分股。

下面来介绍一下3D CNN模型：

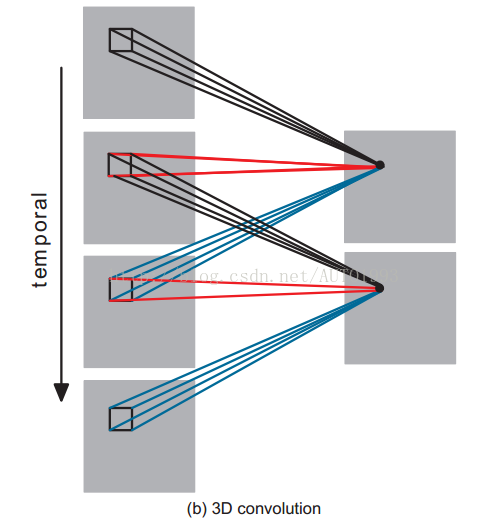
**3D CNN卷积模型**

3D CNN在**视频分类、动作识别**等领域具有优势。

Paper：3D Convolutional Neural Networks for Human Action Recognition

这篇论文可以说是应用3D CNN模型的很重要的一篇参考文献，可以仔细读读。

传统2D识别方式没有考虑到时间维度的帧间运动信息，而3D则加入了时间维度（作为depth），也即几帧连续输入，至于具体输入几帧视情况而定，例如，在表情识别方面大都选择三帧（起始帧，峰值帧，终止帧）。



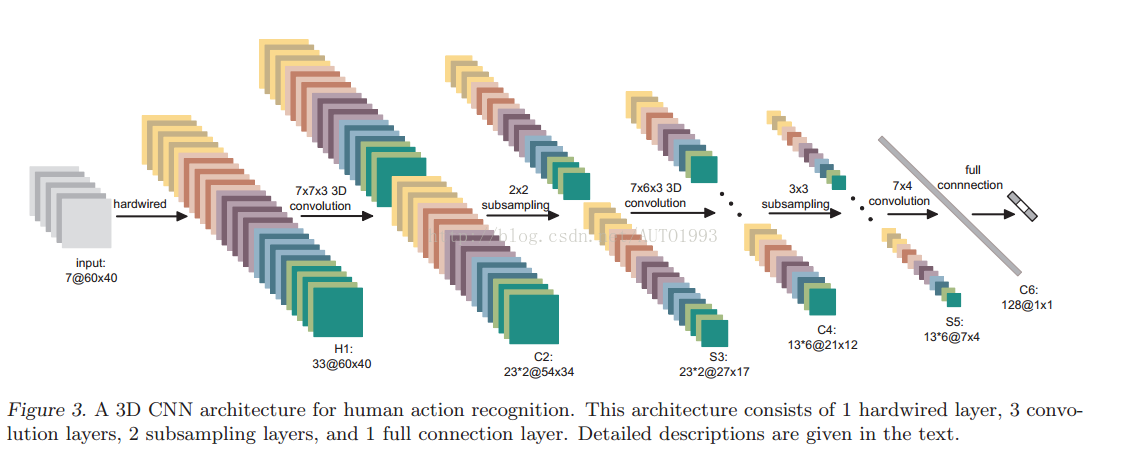
一个卷积map的某一位置的值是通过卷积上一层的三个连续的帧的同一个位置的局部感受野得到的。

上面进行卷及操作的时间维度为3，即对连续的三帧图像进行卷积操作，通过堆叠多个连续的帧组成一个立方体，然后在立方体中运用3D卷积核。

卷积层中每个特征map都会与上一层中多个邻近的连续帧相连，因此捕捉动作信息。

3D卷积核只能从cube中提取一种类型的特征，因为在整个cube中卷积核的权值都是一样的，也就是共享权值，都是同一个卷积核。我们可以采用多种卷积核，以提取多种特征。

对于CNN，一个通用的设计规则是，在后面的层（离输出层近的）特征map的个数应该增加，这样就可以从低级的特征maps组合产生更多类型的特征。



1. 输入层（input）：连续的大小为60\*40的视频帧图像作为输入。

2. 硬线层（hardwired，H1）：每帧提取5个通道信息（灰度gray，横坐标梯度gradient-x），纵坐标梯度gradient-y，x光流optflow-x，y光流optflow-y。

前面三个通道的信息可以直接对每帧分别操作获取，后面的光流（x，y）则需要利用两帧的信息才能提取，因此

H1层的特征maps数量：（7+7+7+6+6=33），特征maps的大小依然是60\*40；

3. 卷积层（convolution，C2）：以硬线层的输出作为该层的输入，对输入5个通道信息分别使用大小7\*7\*3的3D卷积核进行卷积操作（7\*7表示空间维度，3表示时间维度，也就是每次操作3帧图像），同时，为了增加特征maps的个数，在此层采用两种不同的3D卷积层，因此C2层的特征maps数量为(((7-3)+1)\*3+((6-3)+1)\*2)\*2 = 23\*2

这里右乘的2表示两种卷积核。

特征maps的大小为：((60-7)+1)\*((40-7)+1) = 54\*34

4. 降采样层（sub-sampling，S3）：在该层采用max pooling操作，降采样之后的特征maps数量保持不变，因此S3层的特征maps数量为：23\*2

特征maps的大小为：(54/2)\*(34/2) = 27\*17

5. 卷积层（convolution，C4）：对两组特征maps分别采用7 6 3的卷积核进行操作，同样为了增加特征maps的数量，文中采用了三种不同的卷积核分别对两组特征map进行卷积操作。这里的特征maps的数量如下：

从输入的7帧图像获得了5个通道的信息，因此结合总图S3的上面一组特征maps的数量为((7-3)+1)\*3+((6-3)+1)\*2 = 23，可以获得各个通道在S3层的数量分布：

前面的乘3表示gray通道maps数量 = gradient-x通道maps数量 = gradient-y通道maps数量 = (7-3)+1 = 5

后面的乘2表示optflow-x通道maps数量 = optflow-y通道maps数量 = (6-3)+1 = 4

假设对总图S3的上面一组特征maps采用一种7 6 3的3D卷积核进行卷积就可以获得：

((5-3)+1)\*3+((4-3)+1)\*2 = 9+4 = 13；

三种不同的3D卷积核就可以获得13\*3个特征maps，同理对总图S3的下面一组特征maps采用三种不同的卷积核进行卷积操作也可以获得13\*3个特征maps，因此C4层的特征maps数量：13\*3\*2 = 13\*6

C4层的特征maps的大小为：((27-7)+1) \* ((17-6)+1) = 21\*12

6. 降采样层（sub-sampling，S5）：对每个特征maps采用3 3的核进行降采样操作，此时每个maps的大小：7\*4

在这个阶段，每个通道的特征maps已经很小，通道maps数量分布镜框如下：

gray通道maps数量 = gradient-x通道maps数量 = gradient-y通道maps数量 = 3

optflow-x通道maps数量 = optflow-y通道maps数量 = 2

7. 卷积层（convolution，C6）：此时对每个特征maps采用7\*4的2D卷积核进行卷积操作，此时每个特征maps的大小为1\*1，至于数量为128是经验值。

通过多层的卷积和降采样，每连续7帧图像就可以获得128维的特征向量。输出层的单元数与视频动作数是相同的，输出层的每个单元与这128维的特征向量采用全连接。在后面一般采用线性分类器对128维的特征向量进行分类，实现行为识别，3D CNN模型中所有可训练的参数都是随机初始化的，然后通过在线BP算法进行训练。

例子：

2016年Kim等人采用深度学习方法提出了一个基于情感状态的特征表示，使用深度卷及网络（CNN）对例如情绪起止（onset/offset）,情绪峰值（apex）进行编码，同时使用了多种损失函数来促进网络的分类效果。CNN提取图像特征后送入LSTM中学习视频时域上的特征。2017年Peng[27]等人使用3D卷积神经网络（3DCNN），该网络可以适配多种不同帧率的视频。在该模型中，为了得到充分的训练数据，作者同时使用了CASMEI和CASMEII两个数据集来进行网络的训练，同时使用了视频光流信息以丰富输入数据。

M. Peng, C. Wang, T. Chen, G. Liu, and X. Fu, “Dual temporal scale convolutional neural network for micro-expression recognition,” Frontiers in Psychology, vol. 8, p. 1745, 2017.

到此，需要明确**2D与3D的区别**：

这部分详见Note May 14, 2019.pdf文件（这是我在具体进行实验时的笔记，可能笔迹有些混乱，但里面有些在实验中遇到的问题的思考以及解答）。

1. 首先，最最基本的区别在于输入形式（input\_shape）不同，3D CNN的输入要比2D的多出来一维，也即depth，这代表着每次输入几帧（每帧是一张图）。

2. 其次，卷积核形式不同，pool\_size大小不同，都是三维的。

3. 对于股票数据来讲，正常情况下载2D CNN模型中的输入形式为 PastDays×FeatureNums。

所以我们需要先将数据形式转换为适应于3D CNN模型的输入形式，也就是需要多加入一维变量，新加入的这一维变量需要谨慎思考，否则会造成模型学习不到任何内容。

**数据预处理**

需要注意，**数据的预处理**部分是很重要的，请务必不要忽视这部分，接下来将基于我的实验对此部分内容进行详细讲解。

1. 从Yahoo！Finance上获取数据（我在此使用的数据是NYSE Composite成分股）；
2. 数据清洗：
   1. 去除重复数据
   2. 修改错误数据
   3. 填充无效数据
      1. 涨幅超过10%是因为计入了新股的首日涨幅，跌幅超过-10%，可能是由于分红配送等原因。
      2. 在处理中，将涨幅大于10%以及跌幅大于-10%的都归于10%。
      3. 对于当日停牌的数据，去除。
   4. 抽象数据表示
      1. 将所有包含null的行全部删除，当然也可以选择均值填充方式。
   5. 筛选特征值
   6. 分配权重等
      1. 我的代码里现在还没有分配权重，但是可以查看NYSE Composite里面不同成分股的权重，并予以分配。
   7. **数据归一化**
      1. 整体数据归一化（归一化是很重要的，这是为了避免因部分变量的数值过大而对模型准确度造成不必要的影响）。
      2. 经检查后发现，数值还是相差有些大，所以考虑将数值放缩到[0, 255]之间（这只是我个人的尝试，如此处理是为了将股票数据制作成形如图像的形式，每个值对应于图像上不同的像素点）。
      3. 发现放缩过后，预测结果还是不理想，这是由于股票数据多年来价格普遍都是上升的，所以如果按照前7后3这种分法会有训练数据大多数据较小，但在测试数据时却因为数据较大而无法得到理想的效果。

**NOTE：**

之前在数据的预处理时就已经产生了问题，但是没有及时发现，这可能是造成最终结果不佳的原因。

1. 于是在此考虑先对数据进行处理；
2. 再计算change值，因为change值有正有负且值要远小于其他值，所以不能一起处理。【其实可以一起处理，因为在notMNIST数据集中也存在负数值，但是并不影响最终结果。】
3. 将数据放缩到[0, 1]之间，再乘以255，制作成形如图像的形式。
4. 在划分训练集和测试集时不能简单地选取前后7：3分，因为对于股票这类具有时序性的数据而言，后半部分的数据可能和前部分的数据数值相差大，从而造成在训练集上拟合效果良好，但在测试集上拟合效果波动很大的结果。
5. 于是我考虑将所有的数据先拆分为n个batch，然后生成随机数，随机选取m个batch作为训练集，n-m个batch作为测试集，这样打乱处理可以提高预测准确率。

【可以使用pickle模块来存储数据，这样可以方便下次的快捷读取，因为每次拆分数据为batch大小都需要耗费较长时间。】

这个的代码很简单，稍微看一下就懂了，调用起来十分方便迅速。

**具体实验以及结果**

以notMNIST数据集和NYSE Composite（纽约股票交易所综合指数）为例，纽约股票交易所综合指数包括在纽约股票交易所上市所有普通股的指数，旗下并设有四个分组指数：工业、交通、公共事业及金融。指数追踪纽约股票交易所上市普通股市场价值的变动，并消除新上市及摘牌的影响。每种股票市场价值的计算方法为每股价格乘以上市股票数量。

2D CNN notMNIST dataset

X\_train shape: (200000, 28, 28, 1)

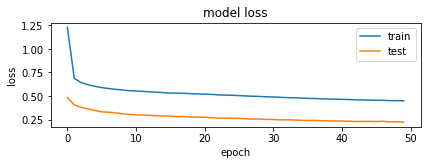
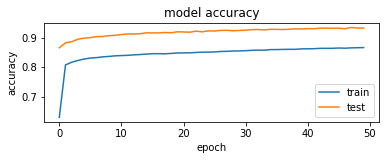
200000

(200000, 28, 28, 1) (200000, 10)

(8733, 28, 28, 1) (8733, 10)

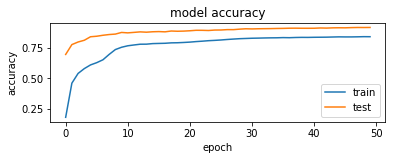
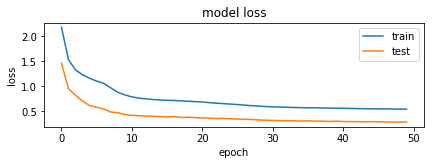
Train on 200000 samples, validate on 8733 samples

Accuracy: 0.9331272185347026 when sample\_size= 200000



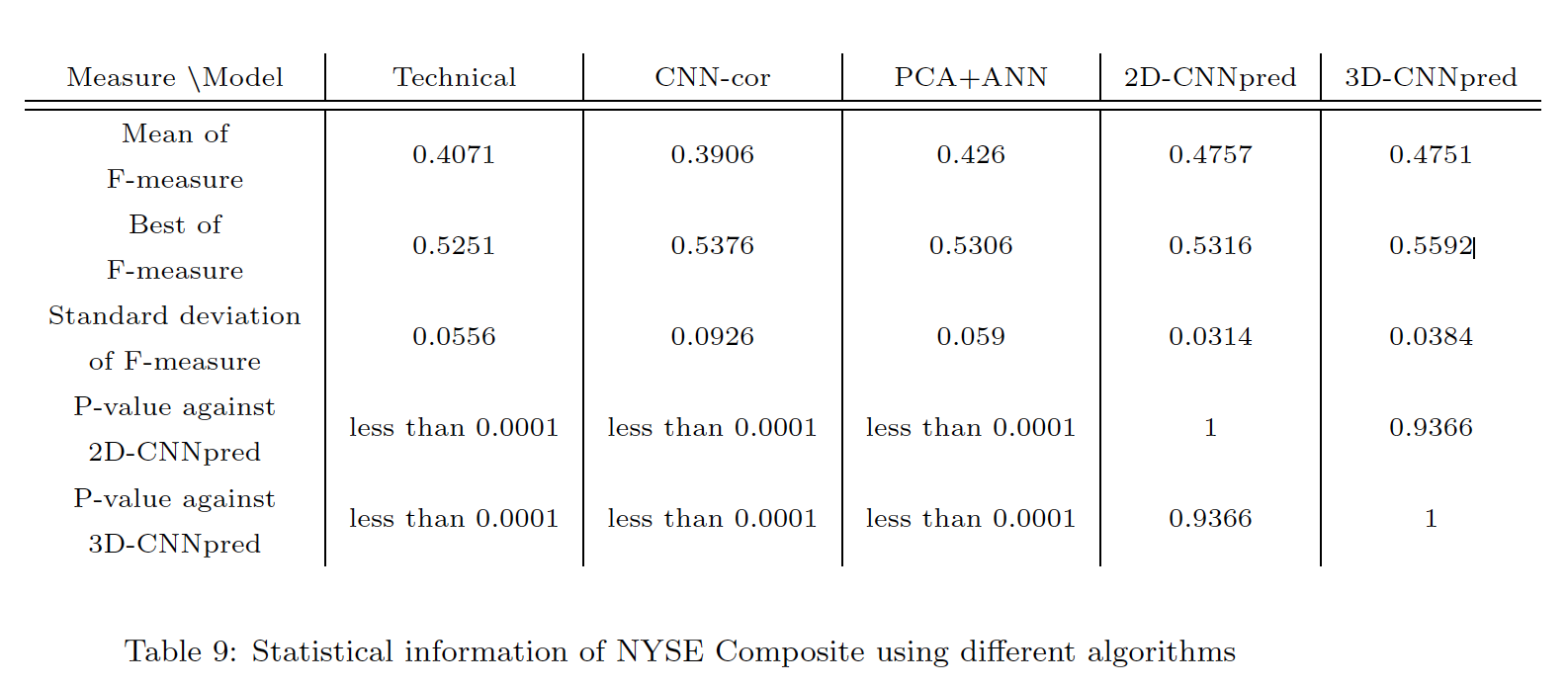
3D CNN notMNIST dataset

Train on 200000 samples, validate on 8748 samples



**NYSE Composite包含的股票：**

共计205支（210支中去除掉5支不符合条件的股票）



这是别的论文中所得到的结果，可以将我们最终得到的结果与之进行比较。

在NYSE Composite成分股共计205支股票上使用2D CNN模型进行拟合预测

1. 模型结构

Layer (type) Output Shape Param #

=========================================================

conv2d\_161 (Conv2D) (None, 3, 4, 32) 320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_161 (MaxPoolin (None, 1, 2, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_321 (Dropout) (None, 1, 2, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_161 (Flatten) (None, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_321 (Dense) (None, 128) 8320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_322 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_322 (Dense) (None, 3) 387

=========================================================

Total params: 9,027

Trainable params: 9,027

Non-trainable params: 0

1. 超参数

|  |  |
| --- | --- |
| PastDays | 6 |
| n\_classes | 3 |
| FeatureNums | 6 |
| batch\_size | 32 |
| num\_epoch | 50 |
| n\_filters | 32 |
| kernel\_size | [3, 3] |
| pool\_size | [2, 2] |

1. 最终结果如下所示：

average F1\_score = 0.6054836778053556

max F1\_score = 0.8755221386800335

min F1\_score = 0.297439225867359

std F1\_score = 0.09300818913173516



average accuracy = 0.6562930760345375

max accuracy = 0.904761910438537

min accuracy = 0.46710526472643804

std accuracy = 0.06410337413796796



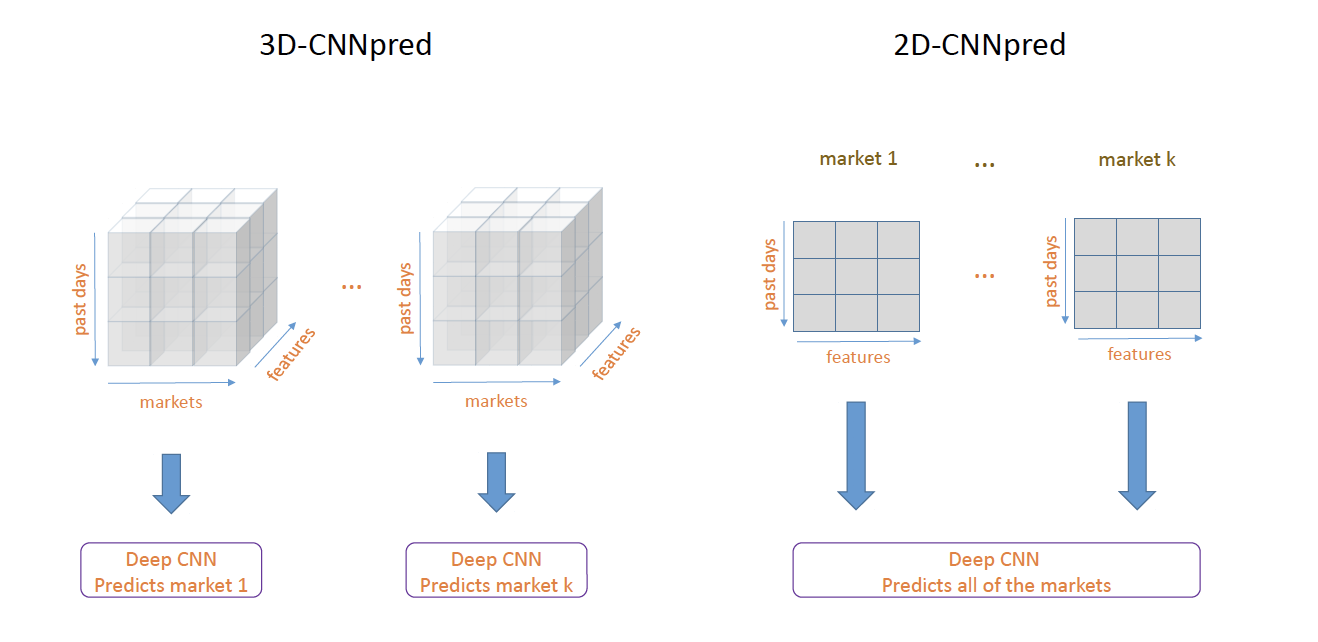
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| F1\_score | |  | Accuracy | |
| 平均 | 0.605483678 |  | 平均 | 0.65629308 |
| 标准误差 | 0.006495974 |  | 标准误差 | 0.00447717 |
| 标准差 | 0.093008189 |  | 标准差 | 0.06410337 |
| 最小值 | 0.297439226 |  | 最小值 | 0.46710526 |
| 最大值 | 0.875522139 |  | 最大值 | 0.90476191 |
| 观测数 | 205 |  | 观测数 | 205 |

在论文中，

在2D CNN模型中，input的形式是PastDays×FeatureNums；

在3D CNN模型中，input的形式是PastDays×5×FeatureNums；

其中FeatureNums作为depth输入。



对于这篇论文中的2D CNN模型是：将所有市场的数据放到一起，共同训练出的模型；

3D CNN模型是针对于每个市场的输入形式都是相同的，但是test集不同；每个输入形式都是由五个市场的数据组成的。

于是接下来，我也使用NYSE Composite中的所有股票数据来共同训练出一个模型。

结果如下：

**三分类**

(402557, 5, 6, 1) (402557, 3)

(172666, 5, 6, 1) (172666, 3)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_2 (Conv2D) (None, 3, 4, 32) 320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 1, 2, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_3 (Dropout) (None, 1, 2, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_2 (Flatten) (None, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 128) 8320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_4 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 3) 387

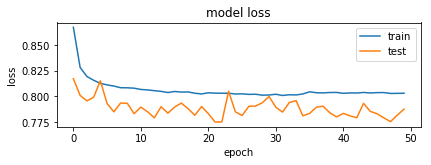
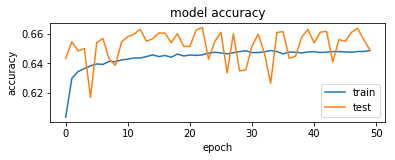
=================================================================

Total params: 9,027

Trainable params: 9,027

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



F1\_score = 0.6293741319457423

confusion matrix

[[79148 2 14821]

[10448 2 3719]

[28972 2 35552]]

Test Score = 0.775121790641456

Test accuracy = 0.6642998621630107

**二分类**，将涨、平归为一类，跌归为一类。

(402557, 5, 6, 1) (402557, 2)

(172666, 5, 6, 1) (172666, 2)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 3, 4, 32) 320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 1, 2, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 1, 2, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 128) 8320

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 2) 258

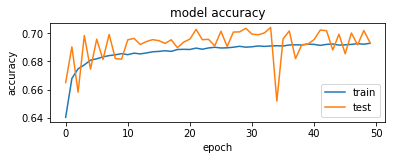
=================================================================

Total params: 8,898

Trainable params: 8,898

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



F1\_score = 0.6955287851325519

confusion matrix

[[35910 33576]

[17501 85679]]

Test Score = 0.5715632119838024

Test accuracy = 0.7041861165514741

可以看出模型在test集上的准确率以及损失值还是较为良好的，接下来可以考虑添加新的变量，或者调整label（在此我使用的label是从PastDays中选出涨、平和跌中天数最多的一天作为标签，例如涨的天数最多，那这个batch的标签就是涨。）

* ATTENTION：如果只将模型应用在一个股票市场上，那么2D的效果要明显优于3D的，而且模型拟合以及预测速度也要明显快于3D的（这部分可以见Note May 14, 2019.pdf，这个文件中有我在notMNIST上运行3D CNN时的详细记录）。