中国石油大学（华东）
CHINA UNIVERSITY OF PETROLEUM

2018—2019 学年第 2 学期
《大数据技术及应用》课程报告

题目：基于 RNN-LSTM 的股票价格预测模型

学　　号 1607040216

专　　业 物联网工程

班　　级 1602 班

姓　　名 吕　涵

得　分	
阅卷人	

2019 年　6　月　1　日

说明

1. 《大数据技术及应用》课程，旨在对学生进行大数据处理能力的综合训练，以提高学生分析问题、建立模型、解决问题能力。所有参加本课程的学生都必须参与算法设计实现、课程报告撰写工作。
2. 《大数据技术及应用》结束后，需要提交**源数据、程序源代码、报告电子版、报告纸质版，压缩后以“学号+班级+姓名”进行命名，材料经班长汇总后交给任课老师**。逾期未提交相关资料者不得参加成绩评定。
3. 指导教师需对学生实践报告填写评语并在相应位置填写成绩并签字。
4. 课程报告要求严格按照本报告模板撰写，条理清晰、内容详尽、论述准确、书写认真。成绩由报告格式设计、内容设计、算法设计、结果描述等几部分组成。
5. 报告格式要求如下：表格、图像进行编号，正文要求宋体、小四，行前缩进 2 字符，间距 1.5 倍行距。其他未尽事宜请参考本科毕业设计报告撰写格式说明，纸质报告要求双面打印。
6. 报告可根据内容适当增加页面，但不宜长篇大论，所写内容应紧扣主题。

题目 基于 RNN-LSTM 的股票价格预测模型

一、研究意义

股票（stock）是股份公司发行的所有权凭证，是股份公司为筹集资金而发行给各个股东作为持股凭证并借以取得股息和红利的一种有价证券，最大特点就是其价格具有变化性。

股票的变化趋势时由空间和时间共同构成。从时间上看，趋势具有周期性高点和低点之间的时间间隔符合一定的规律；从空间上看，趋势具有通道高点的连线和低点的连线光滑有序，两条连线构成的图形或者是平行线，或者呈规律性发散或收敛。时间和空间的加速度各自倾向于一个常数，时间和空间的绝对值可以当量互换，此外，股票变化趋势也具有层次性：上中下级别趋势互相嵌套，三种级别形态具有相似性，各自保持各自的时空加速常数。而且这个常数从绝对值上较为接近，在变化趋势上有规律性。

我们以股票市场的经济学理论为基础，结合当前较为成熟的股票价格变化数学模型即可进行股票走势预测。传统的股票预测方法，如跳空倍数法和消息高开法虽然在证券分析领域已较为成熟，但这只是基于分析人员的历史经验和少量的价格信息做出的行为预测，其方法模型难以具有普适性。此外，传统的预测模型由于数据分析能力的限制，所需分析和运算时间较长，难以实时处理。而且传统模型一般只对未来进行短时预测，难以进行长时间的预测，其应用效果受很大限制。

我们尝试利用循环神经网络（Recurrent Neural Network，即 RNN）的时间序列分析能力，对 APPLE 公司近年股票价格进行数据分析、训练价格模型、做出行为预测。在此之上，考虑到股票时序信息量大、运算工序复杂，我们使用长短期记忆网络（Long Short-Term Memory，即 LSTM）的信息记忆-遗忘功能，先判断信息有用与否在做分析，解决神经网络梯度进行反向传播时，被不必要的信息干扰而带来的问题，从而进一步提高模型精度。

二、数据描述

本实验数据是使用 Python 爬虫获取的 APPLE 公司近年股票价格，来源于全球性的证券与数字货币交易网站 *www.coinmarketcap.com*。爬虫程序基于 Pandas 库的 read_html 函数和 html5lib 的扩展包，爬取结果直接是 dataframe 数据格式。爬取时我们不做限制，尽可能多的爬取最大信息。爬取完成后查看数据可知，该网站的数据起始于 2013 年 4 月 28 日，截止到今日 2019 月 6 月 25 日，每天一条数据，共 2250 条。

每条数据有 7 条标签，分别是日期 Data、当天开盘价 Open、当天最高价 High、当天收盘价 Close、当天交易量 Volume 和股票市场存有量 Market Cap。首先进行数据清洗，我们先遍历所有的 2250 条数据，填充所有缺失的数据点。之后分析数据，根据证券预测的先验经验，我们得出股票的价格涨幅主要受到收盘价 Close 与交易量 Volume 影响，因此我们需要去除不必要的数据列，只分析这两个标签下数据。

初步可视化分析我们发现，前期数据与后期变化幅值非常大，呈现明显的梯度趋势。为了简化训练样本，减少变化幅值大带来的误差，我们只选择 2016 年之后的数据共 1258 条做数据集。数据处理完成后我们根据日期标签进行升序排列，因为我们的模型是基于时序预测，必须保证样本数据的时序正确，按照时间先后从早到晚进行训练。

在构建模型前，我们数据中还存在一个问题：1000 余条数据中，不同标签下的数据度量标准不同，如收盘价 Close 涨幅恒定在数十美元到百余元之间，而交易量 Volume 近亿。若不加干预，则 Volume 的权重值将会极大大于 Close，这对模型的构建带来许多不利影响。因此我们还必须进行数据的归一化。

我们采用最常用的 0~1 归一化，即把各个标签数据映射到 0～1 范围之内处理，这样可保证不同标签数据之间度量标准相同，权重值也相同，不影响模型训练时权重值得调整。

最后我们将数据集剔除标签只保留数值，从 Dataframe 数据类型转为 numpy.array 格式得到数值矩阵样本，之后按照 8：2 的比例将原始样本分离为训练集和测试集，以便检验最终选择最优的模型的性能。

三、模型描述

数据预处理完成后，我们得到 1010 条训练数据和 248 测试数据。首先对训练集进行模型训练，我们主要用到 RNN 与 LSTM 模型。本实验的基本流程是模型设计、构建函数和参数的选择、模型的训练、模型测试和评估。本部分将介绍各流程基本要点。

(1) 模型设计

第一步是设计 LSTM-RNN 模型，在该模型中，我们使用三层 LSTM 层，每层 512 个神经元，未防止过拟合，我们将每层 LSTM 之后设置了比率 0.25 的 Dropout，三层 LSTM 最后进行全链接输出运行结果。

(2) 构建函数和超参选择

第二步是构建模型时，预先选择合适的激活函数、代价函数与优化器，并且定义必要的超参数。查阅资料可知，tanh 函数（双切正切函数）取值范围为[-1,1]，在特征相差明显时效果可观，适用于循环神经网络。tanh 函数与 sigmoid 的区别是 tanh 是 0 均值的，因此循环过程中会不断扩大特征效果，其实际应用效果会比常用的 sigmoid 函数更好。

损失函数我们采用常用回归预测中最为常用的均方误差函数（MSE），计算方法是求预测值与真实值之间距离的平方和。优化器的选择上，我们想尽可能让模型方便实现、减少内存使用且能较好的处理噪音样本，因此 Adam 是最合适的优化算法。

除此优点外，Adma 的步长更新和梯度大小无关，仅由 alpha、beta_1、beta_2 决定步长的理论上限。其对目标函数没有平稳要求，允许损失函数随时间变化，适用与循环神经网络。最重要的一点是它能较好处理稀疏梯度，即在某一区域梯度都是 0 的情况下它不会陷入局部最优解问题。

根据许多现有时间预测模型的先验经验，结合我们股票数据样本的的特点，我们定义了失活率、批处理参数、迭代次数和输入窗口值等超参。在程序中我们将超参数设置为可调变量，以便模型训练效果不理想时进行调参。

(3) 模型的训练

整个模型我们借助 TenseFlow 和 Keras 人工神经网络库搭建，极大简化了模型构建与训练的流程与代码量。首先我们需要初始化 LSTM 的记忆单元

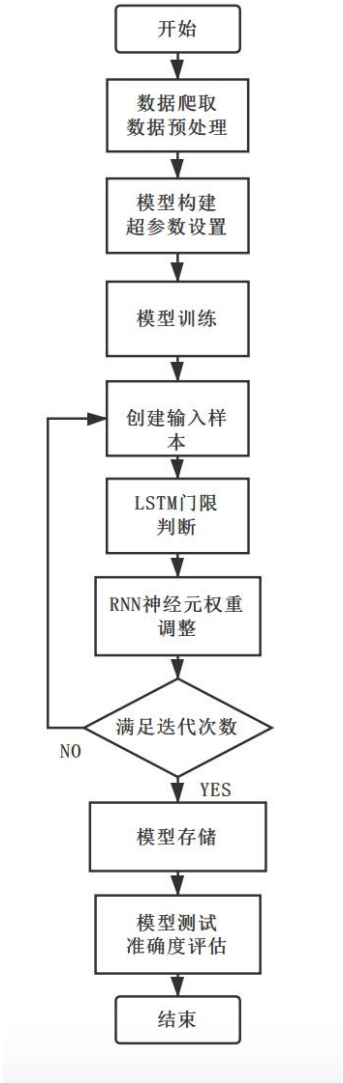
Memory Cell，之后定义随机种子，最后根据我们之前设置的函数和超参数创建模型对象即可。整个模型的训练时间会根据数据量与运行机器的性能差异而不同，训练结束后，为了方便下一步的分析和测试，我们可以将模型以.h5 文件格式保存至本地。

（4）模型测试与评估

本实验中我们刻意将模型训练与测试分离开来，是因为在本地模型训练耗时较大，为了方便调整测试机制，我们先将训练模型存储，再在测试时读取模型文件，免去反复训练的时间。

测试时我们使用训练好的模型对测试集进行拟合，然后与其真实值进行对比计算拟合优度，借助 *matplotlib* 可视化呈现结果，对模型性能进行整体直观的评估。

程序流程图如下：



四、算法实现

本部分按照程序流程顺序进行核心代码描述，主要介绍数据预处理、创建输入输出样本、超参设置、构建 LSTM-RNN 模型、样本初始化、模型训练与模型评估。

(1) #数据预处理与切分

```
def create_model_data(data):#数据预处理
    data = data[['Date']+[coin+metric for coin in ['APPLE_'] for metric
    in ['Close','Volume']]]
    data = data.sort_values(by='Date')
    #只保留股票信息的"日期"、"收盘价格"和"成交量"
    return data
def split_data(data, training_size=0.8):#训练集、测试集数据切分
  #按照 8：2 的比例将原数据集切分为训练集和测试集
  return data[:int(training_size*len(data))],
         data[int(training_size*len(data)):]
```

(2) #创建输入输出样本

```
def create_inputs(data, coins=['APPLE'], window_len=window_len):
    #定义一个窗口长度，将输入样本在 0 和 1 之间进行归一化
    norm_cols = [coin + metric for coin in coins for metric
                in ['_Close','_Volume']]
    inputs = []
    for i in range(len(data) - window_len):
    temp_set = data[i:(i + window_len)].copy()
    inputs.append(temp_set)
    for col in norm_cols:
       inputs[i].loc[:, col] = inputs[i].loc[:, col] /
                               inputs[i].loc[:, col].iloc[0] - 1
    return inputs
```

```
(3) #超参设置
activation_function = 'tanh'#tanh 作激活函数
loss = 'mse'#MSE 作损失函数，
optimizer = "adam"#'adam'作优化器
dropout = 0.25#失活率 0.25，防止过拟合
batch_size = 12#批处理参数 12
epochs = 50#迭代次数
window_len = 7#输入窗口值
training_size = 0.8#训练样本占比

(4) #构建 LSTM-RNN 模型
def build_model(inputs, output_size, neurons,
                activ_func=activation_function,
                dropout=dropout, loss=loss, optimizer=optimizer):
  """模型中使用 3 层 LSTM，每层 512 个神经元，每个 LSTM 层后有 0.25 概率的 Dropout
层，以防止过度拟合（over-fitting）并且每隔一个 Dense 层产生输出"""
    model = Sequential()
    model.add(LSTM(neurons,return_sequences=True,
                   input_shape=(inputs.shape[1], inputs.shape[2]),
                   activation=activ_func))
    model.add(Dropout(dropout))
    model.add(LSTM(neurons, return_sequences=True,
                   activation=activ_func))
    model.add(Dropout(dropout))
    model.add(LSTM(neurons, activation=activ_func))
    model.add(Dropout(dropout))
    model.add(Dense(units=output_size))
    model.add(Activation(activ_func))
    model.compile(loss=loss, optimizer=optimizer, metrics=['mae'])
```

```
    model.summary()
    return model
```

(5) #样本初始化

```
#训练集、测试集去除索引
train_set = train_set.drop('Date', 1)
test_set = test_set.drop('Date', 1)
#训练集初始化
X_train = create_inputs(train_set)
Y_train = create_outputs(train_set, coin='APPLE',window_len=7)
#测试集初始化
X_test = create_inputs(test_set)
Y_test = create_outputs(test_set, coin='APPLE',window_len=7)
#测试集训练集转为 np 矩阵
X_train, X_test = to_array(X_train), to_array(X_test)
```

(6) #模型训练与存储

```
neurons = 512#定义每层 512 个神经元
gc.collect()#LSTM Memory Cell 初始化
np.random.seed(202)#随机种子初始化
#初始化模型架构
apple_model = build_model(X_train, output_size=1,neurons=neurons)
#模型训练
history = apple_model.fit(X_train, Y_train, epochs=epochs,
                          batch_size=batch_size, verbose=1,
                          validation_data=(X_test, Y_test_apple),
                          shuffle=False)
apple_model.save('model.h5')#模型训练存储
```

(7) #模型可视化评估

```
def cal_rr(y0, y):#计算拟合优度
sstot = 0
ave = np.mean(y)
for i in y:
    sstot = sstot + (i - ave) ** 2
ssreg = 0
for i in y0:
    ssreg = ssreg + (i - ave) ** 2
ssres = 0
for i in range(len(y0)):
    ssres = ssres + (y[i] - y0[i]) ** 2
r2 = 1 - ssres / sstot
return r2


plt.plot(real,color='#4169E1')#绘制测试集实际曲线
plt.plot(predict,color='#FA8072')#绘制测试集拟合曲线
r2=cal_rr(real,predict)#计算拟合优度
plt.xlabel('Dates')#X轴为日期
plt.ylabel('Price')#Y轴为当天股票价格
#绘制图表标题
plt.title('Stock Price Prediction on Test Set ',fontsize=16)
#输出拟合优度数值即准确率
plt.text(-2,105,s='准确率'+str(round(r2,3)),
         fontproperties=getChineseFont())
plt.legend(['Actual', 'Predicted'])#绘制图例
#绘制背景网格
plt.grid(color='grey', linestyle=':', linewidth=1,alpha=0.3)
plt.show()#绘制图表
```

五、运行结果及意义说明

本部分将对程序运行中几项关键工作中的结果进行展示和说明，主要包括数据的预览、模型训练结果、模型测试与评估结果等。

（1）数据的预览

对数据的详细处理步骤在本项目第二章中已详细介绍，主要流程包括数据爬取、数据清洗、数据归一化、数据标签筛选等。原始数据(部分)预览如图 5.1 所示。

	A	B	C	D	E	F	G
1	Date	APPLE_Open	APPLE_High	APPLE_Low	APPLE_Close	APPLE_Volume	APPLE_Market Cap
2	2019/6/25	110.072	117.9092	110.072	117.9092	24879684533	2.10E+11
3	2019/6/24	108.5374	110.659	106.1043	110.111	19271652365	1.96E+11
4	2019/6/23	106.9669	112.4614	105.561	108.5537	20998326502	1.93E+11
5	2019/6/22	101.7592	111.5735	101.0704	107.0169	29995204861	1.90E+11
6	2019/6/21	95.2507	101.4456	95.2507	101.4456	20624008643	1.80E+11
7	2019/6/20	92.7306	95.9442	92.3248	95.2716	17846823784	1.69E+11
8	2019/6/19	90.7873	92.9962	90.704	92.7352	15546809946	1.65E+11
9	2019/6/18	93.3547	93.4837	90.049	90.8176	15848210536	1.61E+11
10	2019/6/17	89.8892	94.1641	89.8892	93.2035	15562951919	1.66E+11
11	2019/6/16	88.4144	93.3587	88.1456	89.9449	23348550311	1.60E+11
12	2019/6/15	86.8975	88.5913	86.184	88.3838	18371033226	1.57E+11
13	2019/6/14	82.309	87.1064	81.8339	86.9383	19831162906	1.54E+11
14	2019/6/13	81.4555	83.1157	80.8706	82.3092	18669407147	1.46E+11
15	2019/6/12	79.2543	81.9665	78.6236	81.4586	19034432883	1.45E+11
16	2019/6/11	80.0424	80.2639	77.728	79.2771	17107279932	1.41E+11
17	2019/6/10	76.9228	80.3191	75.8673	80.0033	18689275117	1.42E+11
18	2019/6/9	79.4967	79.7597	75.8322	76.8808	16610726547	1.36E+11
19	2019/6/8	80.3677	80.7689	78.3761	79.5413	16522722810	1.41E+11
20	2019/6/7	78.269	81.2615	77.8837	80.4395	19141423231	1.43E+11

图 5.1 原始数据(部分)预览

经过数据处理后我们可划分为 8：2 的训练集与测试集，对于训练集与测试集的部分数据预览如图 5.2 所示。

序号	价格	序号	价格	序号	价格	序号	价格
0	64.8235	10	63.1761	20	64.0922	30	44.5187
1	64.8716	11	63.7778	21	64.1127	31	46.0217
2	64.7574	12	63.8844	22	63.7127	32	43.6594
3	64.9584	13	63.6126	23	63.5949	33	43.4711
4	64.7629	14	63.7613	24	57.3835	34	38.8076
5	64.7475	15	64.1966	25	56.4803	35	40.0997
6	64.8038	16	64.6101	26	55.7555	36	37.7913
7	64.8639	17	65.3014	27	55.5433	37	38.2072
8	63.3263	18	64.5372	28	56.2354	38	42.5742
9	63.3427	19	63.8562	29	48.7149	39	42.7885

图 5.2 训练集与测试集的部分数据预览

(2) 模型训练结果

我们的模型基于 Keras 库来进行构建和训练，Keras 有两种类型的模型，分为序贯模型(Sequential)和函数式模型(Model)，我们所选用的函数式模型应用更为广泛，序贯模型是函数式模型的一种特殊情况。

我们将模型以 h5 格式存储到本地，模型文件大小 60 余 M。再次使用时使用 Keras 库内指令可直接调用模型，我们打印该模型的概述如图 5.3 所示。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 7, 512)            1062912
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 512)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 7, 512)            2099200
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 512)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 512)               2099200
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 5,261,825
Trainable params: 5,261,825
Non-trainable params: 0
_________________________________________________________________
```

图 5.3 RNN-LSTM 模型的概述

图中可看到,模型共有三层 LSTM，每层 512 个神经元，每层 LSTM 之后设置了比率 0.25 的 Dropout。三层 LSTM 最后进行全链接,启用激活函数输出运行结果。整个模型中共有 5,261,858 个参数,这些参数都是根据 RNN 的时间序列和 LSTM 的记忆-遗忘机制自动调整的，无需我们人工参与，这样随减少了模型的透明度，但可以极大地提升工作效率和准确度。

(3) 模型测试与评估结果

测试时我们使用训练好的模型对测试集进行拟合，然后与其真实值进行对比计算拟合优度，借助 *matplotlib* 可视化呈现结果，对模型性能进行整体直观的评估，可视化结果如图 5.4 所示。

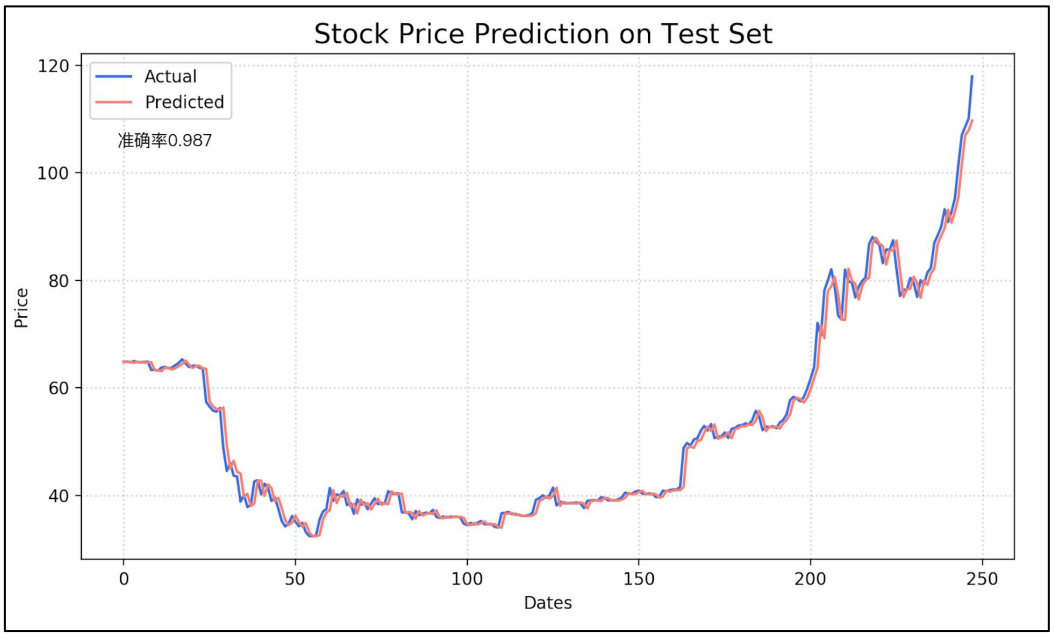


图 5.4 模型测试与评估结果

在可视化呈现的图中，我们绘制了红蓝两条曲线，根据左上角的图例指示可知，蓝色曲线是测试集真实数据而红色曲线是模型对测试集的拟合数据。为了方便图像显示和观察，我们将横坐标的日期按时间先后顺序简化为数字序列。从图上看，测试集的真实数据与拟合数据贴合程度极高，变化趋势基本一致，仅有细微差别。

根据统计学公式：

$$R^2 = 1 - \frac{Res}{Tot} = 1 - \frac{\sum_{i=1}^{n}[y_i - y_i^{'}]^2}{\sum_{i=1}^{n}[y_i - \overline{y}]^2}$$

可计算拟合优度，四舍五入得 0.987，如图中图例下方所显示。故本 RNN-LSTM 模型在测试集上的准确度约为 98.7%，完全符合预期。说明该模型的设计方向和函数、参数选择基本正确，训练过程也比较严谨客观。

六、总结

本次实验中，我们利用循环神经网络（RNN）对 APPLE 公司近年股票价格数据训练模型、做出行为预测。其中的关键核心是使用长短期记忆网络（LSTM）借助其信息记忆-遗忘功能，判断信息的分析价值，避免了神经网络梯度进行反向传播时被不必要的信息干扰，从而进一步提高了模型精度。

在模型设计初期，面对模型函数参数的选取问题，我们查阅了大量的资料，参考了许多成熟模型的经验，学习到了许多重点要点。例如，我们详细对比了 sigmoid、Relu、tanh、softmax 等函数的优缺点之后，结合本例的时序分析需求，选择了最为适合的 tanh 做激活函数。在优化器和损失函数的选取上，同样也进行了对比选取，学习了许多先进函数的工作原理，最终找出了最为适合的作损失函数 MSE 和优化器 adam。

经过多次的参数调整，我们确定下来最为合适的失活率、迭代次数和窗口值等，最终训练的模型对测试集的检验中得到高达 98.7%的准确率，这说明该模型可完全适用于 APPLE 公司股票今年的变化趋势预测，达到了项目预期，弥补了传统预测方法的不足。

然而我们模型的预测行为只是基于完全的数值因素为既定的前提条件为基础的，但在实际情况下，股票价格的变化与国家的宏观经济发展、法律法规的制定、公司的运营、股民的投资等等都有关联，因此我们的模型在经济学上的理论基础还是有所欠缺的。

本实验的模型的 RNN-LSTM 的股票价格预测模型在训练时是相对不透明的，虽然可免去我们对每个参数进行干预的繁琐，但同时也失去了我们对每层每个神经元的分析能力。在下一步的优化中，可以考虑借助可视化手段，将模型进行形象化的可视表示，有利于我们进一步了解模型的计算机制和权值调整。

通过本次实验我收获颇丰，通过详细的学习 RNN 和 LSTM 的基本理论和 Python 语言实现方法，熟悉了深度学习中人工神经网络的训练和使用，也为今后的学习和工作打下了坚实基础。

学生签字:

指导教师评语

指导教师签字:

年　　月　　日