中图分类号：TB114.3

论 文 编 号 ：10006ZY1714116

北京航空航天大學

专业硕士学位论文

基于 LSTM 循环神经网络的时间序列预测研究

作者姓名　胡树立

学科专业　工业工程

指导教师　张叔农 副研究员

培养院系　可靠性与系统工程学院

Study on time series prediction based on LSTM recurrent neural network

A Dissertation Submitted for the Degree of Master

Candidate：Shuli Hu

Supervisor：Associate Professor Shunong Zhang

School of Reliability and Systems Engineering

Beihang University, Beijing, China

中图分类号：TB114.3

论 文 编 号 ： 10006ZY1714116

专 业 硕 士 学 位 论 文

基于 LSTM 循环神经网络的时间序列预测研究

作者姓名　胡树立　　申请学位级别　硕士

指导教师姓名　张叔农　　职　　称　副研究员

学科专业　工业工程　　研究方向　基于数据驱动的 PHM

学习时间自　2017 年　9 月　1 日　起至　2020 年　1 月　15 日止

论文提交日期　2019 年　12 月　12 日　论文答辩日期　2019 年　11 月　29 日

学位授予单位　北京航空航天大学　学位授予日期　年　月　日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中做出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：_____ 日期： 年 月 日

学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名：_____ 日期： 年 月 日

指导教师签名：_____ 日期： 年 月 日

摘 要

时间序列数据（Time-series Data）是在不同时间上收集到的数据，用于所描述现象随时间变化的情况。这类数据反映了某一事物、现象等随时间的变化状态或程度。时间序列数据广泛存在于各行各业，如金融，天气，机械，电子，航空航天等领域，数据对于行业的发展非常重要。因此时间序列预测技术也成为了研究的热点。

传统的时间序列预测方法主要是在确定时间序列参数模型的基础上求解出模型参数，并利用求解出的模型完成预测工作。传统的时间序列预测方法非常依赖参数模型的选择，能否正确选择参数模型在很大程度上决定了预测结果的准确率。对于数据量较多的高维数据，传统的模型很难有效地进行分析预测，因此诞生了很多基于机器学习的时间序列预测方法。在众多机器学习的方法中，循环神经网络（Recurrent Neural Network，RNN）的应用最为广泛，在 RNN 的众多变体中，长短期记忆网络（Long Short Term Memory，LSTM）以其特殊的“门”结构的设计，解决了 RNN 长期记忆能力不足和梯度消失、爆炸的问题，在时序预测领域如金融、交通、天气等方面得到了广泛的应用，但 LSTM 在可靠性领域的相关研究还较少。

本文的具体研究内容如下：

（1）利用主成分分析法（Principal Component Analysis，PCA）对数据进行特征提取。

针对高维的监测数据，为了方便后续的模型训练预测，提高预测和评估效率和准确度，利用 PCA 算法对高维数据进行降维处理，提取出数据的主要特征。

（2）利用多层 LSTM 神经网络模型对高维时间序列数据进行预测分析。

对于高维的海量的传感器数据，利用 LSTM 及其变体分别得到传感器数据和剩余使用寿命 RUL（Remaining Useful Life）的映射，搭建预测模型，并在数据集上进行验证。

（3）研究一种基于贝叶斯理论的模型优化方法。

预测模型参数众多，手动调参是一项非常繁琐的工作，在自动调参方法里，网络搜索比较耗费时间，随机搜索往往准确度不足。本文采用的贝叶斯方法与其他调参方法最大的不同是会充分利用“历史信息”，即会考虑采用过的参数组合，在参考过去结果的基础上决定下一组超参数的选择。并在模型上进行验证。

（4）利用多层 Conv-LSTM 神经网络模型对高维时间序列数据进行预测分析。。

普通的 LSTM 模型只考虑了不同时刻之间的状态转变及输入输出的映射关系，没有

考虑同一时刻数据的空间特征，在 LSTM 的诸多变体中，Conv-LSTM 的核心本质还是和 LSTM 一样，将上一层的输出作为下一层的输入。不同的地方在于 Conv-LSTM 加入了卷积操作之后，不仅能够得到时序关系，还能够像卷积层一样提取空间特征，这样 Conv-LSTM 就可以同时提取时间特征和空间特征（时空特征），并且状态与状态之间的切换也换成了卷积运算。本文利用 Conv-LSTM 模型对时序数据进行分析预测，可同时提取数据的时间和空间特征，提高预测的精度和效率。

关键词：主成分分析，长短期记忆神经网络，Conv-LSTM，贝叶斯优化

Abstract

Time series data widely exists in all walks of life, such as finance, weather, machinery, electronics, aerospace and other fields, data is very important for the development of the industry. Therefore, time series prediction technology has become a research hotspot.

PHM in forecasting method based on data driven does not need to know about the object of study of the physical background, only the researchers need to develop a set of mature mathematical modeling scheme, data driven method of prediction accuracy is hard to reach the solution of physical modeling, but it avoids the difficulty of physical modeling for complex equipment, thus has better universality. Research in recent years has focused on the direction of combining data-driven methods with popular artificial intelligence technologies. The specific research contents of this paper are as follows:

(1) A feature extraction method based on PCA.

For high-dimensional monitoring data, in order to facilitate subsequent model training and prediction and improve efficiency and accuracy, PCA algorithm was used to conduct dimensionality reduction processing on the data and extract the main features of the data.

(2) A residual life prediction method based on multilayer LSTM and residual life prediction method based on multilayer Conv-LSTM.

For the massive high-dimensional sensor data, LSTM and its variants were used to obtain the mapping of sensor data and RUL of remaining life respectively, build the prediction model, and verify it on the data set.

(3) A model optimization method based on bayesian theory.

The prediction model has a large number of parameters. On the basis of referring to a variety of parameter optimization theories, the automatic model optimization method based on bayesian theory is used to optimize the model parameters and verify the model.

(4) A multi-layer conv-LSTM neural network model to predict and analyze high-dimensional time series data..

The common LSTM model only considers the state transition between different times and the mapping relationship between input and output, and does not consider the spatial characteristics of the data at the same time. In this paper, conv LSTM model is used to analyze and predict the time series data, which can extract the time and spatial characteristics of the data at the same time, and improve the accuracy and efficiency of prediction

Key words: PCA, LSTM, Conv-LSTM, Bayesian optimization

,

目 录

摘 要.....I
Abstract.....III
目 录.....IV
图目录.....VII
表目录.....IX
第一章 绪论.....1
1.1 研究背景.....1
1.2 研究意义.....2
1.3 国内外研究现状.....3
1.3.1 时间序列数据.....3
1.3.2 时间序列预测技术.....6
1.3.3 深度学习在时间序列预测的应用.....10
1.3.4 数据预处理方法.....12
1.3.5 目前存在的问题.....14
1.4 软硬件平台.....14
1.4.1 硬件平台.....14
1.4.2 软件平台.....15
1.5 模型所用数据.....16
1.6 论文主要内容与结构安排.....16
第二章 数据预处理.....18
2.1 数据介绍.....18
2.1.1 NASA 发动机数据.....18
2.1.2 数据详细信息.....18
2.2 基于 PCA 的特征提取.....20
2.2.1 PCA 原理.....20
2.2.2 PCA 特征提取步骤.....23
2.3 去中心化和归一化.....27
2.4 本章小结.....29
第三章 基于 LSTM 的 RUL 预测.....30

3.1 LSTM 神经网络 30
3.1.1 网络层 30
3.1.2 LSTM 神经元结构 31
3.1.3 LSTM 公式 32
3.2 模型搭建 33
3.3 案例一 36
3.3.1 数据 37
3.3.2 模型一 37
3.3.3 模型二 38
3.3.4 模型三 39
3.4 案例二 41
3.4.1 数据 41
3.4.2 预测过程与结果 42
3.5 贝叶斯自动调参 47
3.5.1 贝叶斯调参步骤 47
3.5.2 现有结果分析 49
3.6 本章小结 52
第四章 基于 Conv-LSTM 的 RUL 预测 53
4.1 卷积 LSTM 53
4.1.1 LSTM 常见变体 53
4.1.2 Conv-LSTM 原理 55
4.2 RUL 预测 56
4.2.1 数据 56
4.2.2 模型搭建 57
4.2.3 两种 LSTM 模型的比较 59
4.3 本章小结 60
总结与展望 61
总结 61
展望 62
参考文献 63

附录 68
附录 1 LSTM 模型代码 68
附录 2 贝叶斯调参代码 80
附录 3 PCA 代码 81
附录 4 Conv-LSTM 模型部分代码 83
攻读硕士学位期间取得的学术成果 84
致谢 85

图目录

图 1 一组时间序列数据.....3
图 2 时序数据结构.....4
图 3 论文结构.....17
图 4 模拟发动机的基本片段.....18
图 5 原始数据可视化.....20
图 6 基于 PCA 的特征提取算法.....21
图 7 降维归一化的数据可视化图.....28
图 8 2 层神经网络的体系结构.....30
图 9 神经网络迭代过程.....31
图 10 LSTM 神经元结构.....31
图 11 学习率在训练过程中的衰减曲线.....34
图 12 模型结构.....36
图 13 LSTM epoch loss 变化图.....37
图 14 LSTM 预测结果（部分样本）.....38
图 15 LSTM epoch loss 变化图.....38
图 16 LSTM 预测结果（部分样本）.....39
图 17 LSTM epoch loss 变化图.....40
图 18 LSTM 预测结果（部分样本）.....40
图 19 LSTM 模型结构.....42
图 20 训练过程.....43
图 21 训练过程中误差变化.....44
图 22 模型在 4 个样本的预测结果.....45
图 23 55 号电池预测结果.....45
图 24 55 号电池预测误差变化.....45
图 25 54 号电池预测结果.....46
图 26 54 号电池预测误差变化.....46
图 27 53 和 56 号电池预测结果与误差变化.....47
图 28 学习率的取值.....49
图 29 学习率和 mae 散点图.....50
图 30 LSTM epoch MSE 变化图.....50
图 31 LSTM 预测结果（部分样本）.....51
图 32 Conv-LSTM 的结构.....53
图 33 Conv-LSTM 做 RUL 预测.....56

图 34　Conv-LSTM 模型结构.....57
图 35　训练和预测过程中误差变化.....58
图 37　模型在 4 个样本的预测结果.....58

表目录

表 1 常见数理统计模型.....7
表 2 统计模型的局限.....8
表 3 常见机器学习方法.....8
表 4 机器学习方法特点.....9
表 5 LSTM 解决的问题和方式.....11
表 6 LSTM 的应用.....11
表 7 数据中可能存在的问题.....12
表 8 数据清理方法.....12
表 9 硬件配置.....14
表 10 3 号发动机的部分参量数据（前 10 行）.....19
表 11 剩余使用寿命的取值依据.....19
表 12 PCA 特征提取步骤.....23
表 13 3 号发动机的矩阵 X.....23
表 14 X 零均值化的矩阵.....24
表 15 协方差矩阵 C（24*24）.....24
表 16 特征值.....25
表 17 特征矩阵.....25
表 18 最大的 10 个特征值及行号.....26
表 19 特征矩阵 P.....26
表 20 降维后的矩阵 Y（7*150）.....26
表 21 降维后的新数据.....27
表 22 常用优化器及其特点.....34
表 23 5 号锂电池部分数据.....41
表 24 预处理后的数据.....42
表 25 不同轮次的误差比较.....44
表 26 LSTM 主要变体.....54
表 27 两种 LSTM 模型对比.....59

第一章　绪论

1.1 研究背景

时间序列数据广泛存在于各行各业[1]，如金融，天气，机械，电子，交通，医疗，物流，餐饮，基建，互联网，航空航天等领域，愈来越多的数据在生产和生活中产生。数据就是隐藏的财富，数据对于行业，企业，个人的发展非常重要。因此时间序列预测技术也成为了研究的热点[2]，广泛应用于各种实际生产和技术中。

时间序列数据的研究和应用取得了很多成功的案例[3]，比如在机器翻译领域，IBM 公司做了大量的研发投入，试图从语义分析的角度实现机器翻译，效果并不尽如人意，与此相反，Google 公司从数据角度研究实现了在线机器翻译，并且效果良好。时间序列数据的研究不仅在机器翻译的领域应用实现了应用，在其他领域如金融数据，天气预报，交通流量预测，物流速度评估，网络流量预测等也取得了不错的成绩。时序数据通常具有数量较多，维度较高，时效性等特点[4]。因此，研究如何利用时间序列数据挖掘技术对时间序列数据进行有效的预测分析具有重要的意义。

传统的时间序列预测方法主要是在确定时间序列参数模型的基础上求解出模型参数，并利用求解出的模型完成预测工作。例如 ARIMA 模型（Autoregressive Integrated Moving Average model，自回归差分滑动平均模型），对于给定的时间序列，首先确定适当的 p，d，q 值（p 为自回归项数，d 为差分阶数，q 为滑动平均项数）；然后通过最有效的方法估计出模型中具体的参数值；最后检验拟合模型的适当性，并适当的改进模型。传统的时间序列预测方法非常依赖参数模型的选择，能否正确选择参数模型在很大程度上决定了预测结果的准确率。对于数据量较多的高维数据，传统的模型很难有效的进行分析预测，因此诞生了很多基于机器学习的时间序列预测方法。

时间序列数据预测工作本质上与机器学习方法分类中的回归分析之间存在着紧密的联系。经典的支持向量机（Support Vector Machine, SVM）、贝叶斯网络（Bayesian Network，BN）、矩阵分解（Matrix Factorization，MF）和高斯过程（Gaussian Process, GP）在时间序列预测方面均取得了不错的效果。早期的人工神经网络（Artificial Neural Network，ANN）也被用来获取时间序列中长期的趋势。随着深度学习的崛起，其也可以被看作实现时间序列预测的有效工具。

深度学习的本质是通过构建具有多层神经网络层的模型，训练大量的数据，来学习更多有用的特征，最终提高预测，分类或聚类的准确性。因此，“多层神经网络模型”是

手段，“提取数据特征”是目的。与传统的浅层学习不同，深度学习的区别在于:1)模型结构通常有较大的深度，通常有 4 层、5 层甚至 10 层的网络层（特制隐含层）[5];2)明确强调学习数据特征的重要性，即通过逐层特征变换，将数据样本在原始空间中的特征提取出来，转化为新的特征空间，从而降低了预测，分类或聚类的难度。相较于传统的人工规则构造特征的方法，使用大数据来学习数据特征，能全面准确地表征数据丰富的内在信息。

在众多深度学习模型中，循环神经网络（Recurrent Neural Network，RNN）将时序的概念引入到网络结构的设计中，使其在时序数据的分析中表现出更强的适应性。在众多 RNN 的变体中，长短期记忆神经网络（Long Short Term Memory，LSTM）由于其特殊的“门”设计，不仅继承了 RNN 的强大的时序数据处理能力，而且弥补了 RNN 的梯度消失和梯度爆炸、长期记忆能力不足等问题，使得 LSTM 能有效的利用长距离的时序信息。LSTM 模型在不同领域的时序数据研究中已有不少成功的应用案例，包括文字语言相关的语音识别、语言建模、机器翻译；多媒体相关的音频和视频数据分析、图片标题建模；道路运输相关的交通流速预测；生物基因相关的蛋白质二级结构序列预测。然而，在可靠性与系统工程领域，LSTM 模型的应用非常有限，特别是对于剩余使用寿命时间序列预测这一研究问题，目前的相关研究比较少。

本论文研究正是在基于上述背景的前提下，研究 LSTM 循环神经网络在时间序列预测的应用方法，并给出实际工程案例数据集的测试，评估本文中的方法效果。

1.2 研究意义

系统运行产生的传感器数据通常数据量级较大，参数较多，在系统的 RUL 预测分析问题上，传统的线性分析方法无法处理此类数据。

针对上述问题，本文提出在基于 LSTM 循环神经网路的时序数据预测方法，研究意义主要有：

1) 利用 LSTM 循环神经网络方法找到高维传感器数据同剩余使用寿命之间的映射，可以预测当前状态下系统的 RUL，预测精度高，相比物理建模的方法，模型的泛化能力强，普适性更高，降低了寿命预测的难度，只要有一套成熟的数据预测方案，可解决多种对象的寿命预测问题。

2) 为系统有计划的维修，更新部件提供参考，有了维修更新计划，可以减少因故障原因造成的额外成本，保证生产的正常有序进行。

1.3 国内外研究现状

1.3.1 时间序列数据

直观上讲，时间序列就是一组按照时间顺序排列的数值集合。图 1 就是一个典型的时间序列。

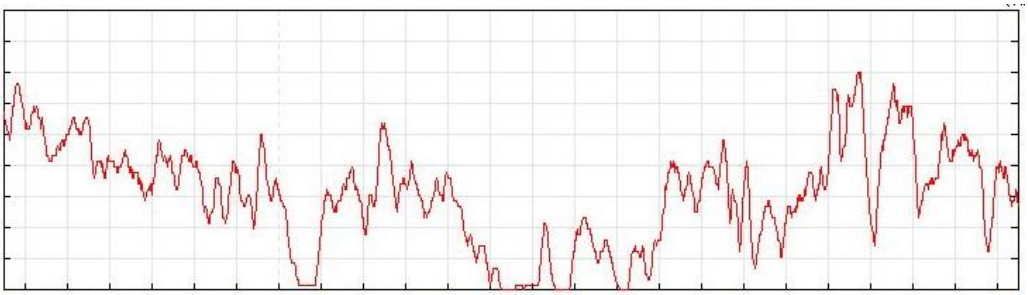


图 1 一组时间序列数据

时间序列（Time Series）:一个时间序列$T=t_1,t_2,\ldots\ldots,t_n$为 n 个数值的有序集合，其中 1，2，…，n 为长度为 n 的时间节点。

对时间序列的概念可以从三个方面理解：从统计意义上讲，时间序列指时间序列数据，即某个指标在不同时间点上的数值根据时间的先后顺序排列而成的数列；从数学意义上讲，时间序列是随机过程的一个特例，即由有序的随机变量组成的序列；从系统意义上讲，时间序列是某一系统在不同时间上的响应，即系统在不同时刻的状态值。基于不同的理解，时间序列广泛存在于不同领域。

通常，在数据挖掘中对于整个时间序列的分析比较困难，而是通过研究时间序列的局部子段来了解整个时间序列的信息。这些局部子段被称为时间序列的子序列。

时间序列子序列（Time Series Subsequence）：给定一个长度为 n 的时间序列 T，长度为 m 的子序列 C 为 T 的一个连续采样，即$C=t_p，t_{p+1},\ldots，t_{p+m-1}$,其中 p 为采样的起点，满足 1<p<n-m+1。

时间序列数据和普通静态数据的一个重要区别就是时间序列具有时间特性。时间序列的时间特性包括：

1）时间序列数据是有时间相关性的，即后一时候的值是受前一时刻值的影响，而普通静态数据和时间无关。

2）在时间序列的演变过程中，对于当前时间上数据的影响是随着时间的前移越来越小的，即在进行时间序列数据挖掘时，越接近当前时间的区段影响越大，越反映最新的时间序列信息；而远离的区段则影响较小，其在时间序列数据挖掘中的价值也越小。

3）随机抽样不适合于时间序列数据。随机抽样不能反映数据按照时间的变化，甚至会将数据的时间顺序打乱。

4）数据随着时间的前进不断产生，时间序列数据挖掘应尽可能采用扫描次数少和运算量小的算法。

因此，不论在时间序列数据挖掘，还是进行时间序列的建模时都必须将时间序列的特性考虑在内，尽量保留其时间特征。

1.3.1.1 时间序列的分类

从不同的研究角度，时间序列有不同的分类。

1）按照时间序列的变量数目不同，可以分为单变量时间序列和多变量时间序列。单变量时间序列是指只一个变量的数值随时间变化所得到的序列；多变量时间序列是指多个变量的数值随时间变化所得到的多维序列。多变量时间序列的定义如下：

多变量时间序列（Multivariate Time Series）:一个大小为 d*n 的多变量时间序列是变量数目为 d 个长度为 n 的时间序列的集合。

多变量时间序列可以用下面的形式表示：

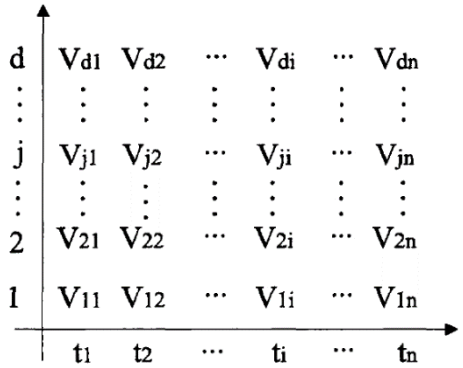


图 2 时序数据结构

其中，V_{ji} 表示第 j 个变量在时间 t_i 上的值。当 d=1 时，就退化为单变量时间序列；当 d≥2 时，则表示多变量时间序列。现实应用中，多变量时间序列越来越多，针对它的研究也越来越重要。

2）按照时间的连续性，可以分为连续型时间序列和离散型时间序列。如果观测值是连续获得的，则为连续型时间序列；如果观测值是离散获得的，则为离散型时间序列。在现实应用中，一般研究的是离散型时间序列。连续型时间序列可以通过等间隔采样或者每隔一段时间采集某变量的累积量来转换成离散型时间序列。

3）按照时间序列的统计特征，可以分为平稳时间序列和非平稳时间序列。平稳时间序列又分为严平稳（狭义平稳）和弱平稳（广义平稳）。严平稳是指对于不同的时间，时

间序列都有相同的概率分布现实应用中严平稳时间序列极少；弱平稳是指时间序列的均值为常数，协方差与时间点无关而与时间的间隔有关。不满足平稳性条件的为非平稳时间序列。在传统的时间序列分析方法中，非平稳时间序列首先要转换为平稳时间序列，而在时间序列数据挖掘中，对时间序列的平稳性一般没有要求。

4）按照时间序列的分布规律，可以分为高斯型时间序列和非高斯型时间序列。符合高斯分布（正态分布）的时间序列为高斯型时间序列，否则为非高斯型时间序列。对于一些非高斯型时间序列，可以通过适当的变化近似地看做高斯型时间序列。

5）按照时间序列的数值类型，可以分为数值型时间序列和类别型时间序列。数值型时间序列是指观测值为数值型的时间序列，比如温度{22，20，21，22，23，25}；类别型时间序列是指观测值为名词型的时间序列，比如{天气，晴天，阴天，小雨，多云}。在时间序列数据挖掘中，数值型的时间序列常用于预测或回归研究，类别型时间序列用于分类研究。

1.3.1.2 时间序列的属性

时间序列数据是按时间顺序排列的一系列观测值。与一般的定量数据不同，时间序列数据包含时间属性，不仅要表达数据随时间变化的规律，还需表达数据分布的时间规律。早期，人们将时间序列数据绘制在图纸上，以图形可视化的方法来发现时间序列数据的规律。计算机技术发展以来，涌现出许多基于时间序列数据的研究[6]，例如相似序列搜索、降维、聚类、分类、模式分析、预测等。

时间序列数据的定义包含两个方面，一是数据与时间密切相关，并随时间变化；二是数据按时间的先后顺序排列。因而，时间序列数据的特征包含以下两个方面：

1）时间属性[7]

时间具有特殊的语义结构，经过人为抽象划分为不同层次的时间尺度，例如分、时、天、周、月等。各层次间的包含关系有的是规则的（例如 60 分钟为一小时，7 天为一周等），有的是不规则的（例如一个月可以是 30 天或 31 天）。时间隐含内在的周期性特征，例如季节的更迭。时间还具有确定性和不确定性的特征，例如列车经过站点的时间有一定的规律，但也可因特殊情况晚点，导致时间不确定。

2）数据属性[7]

按统计尺度分为定性和定量特征；按参照标准可分为非空间和空间特征；按变量个数分为单变量和多变量特征。

1.3.1.3 时间序列的应用

时间序列数据的研究和应用取得了很多成功的案例[1]，比如在机器翻译领域，IBM 公司做了大量的研发投入，试图从语义分析的角度实现机器翻译，效果并不尽如人意，与此相反，Google 公司从数据角度研究实现了在线机器翻译，并且效果良好。时间序列数据的研究不仅在机器翻译的领域实现了成功的应用，在其他领域如金融数据，天气预报，交通流量预测，物流速度评估，网络流量预测等也取得了不错的成绩。

1.3.2 时间序列预测技术

时间序列的预测是通过对过去的和现在的已知时间序列进行分析来推断以后的未知时间序列。对于数值型时间序列，一般称为回归或预测，预测的效果通过绝对误差、相对误差、均方根误差等来进行评价；对于类别型时间序列，预测任务主要是对时间序列的分类，预测效果通过分类准确率等来进行评价。

1.3.2.1 单变量时间序列的预测

单变量时间序列预测就是根据自身的已知数据，通过建模等方法预测未来时间的数据。其数学描述如下：

给定一个已知的时间序列$T=\{T_1,\ T_2,\ldots\ldots,\ T_n\}$，其中 n 为时间序列的长度，对未来时间序列 T′ 的预测是

$$T=\{T_1,\ T_2,\ldots\ldots,\ T_n\}\rightarrow T'=\{T_{n+1},\ T_{n+2},\ldots\ldots,\ T_{n+k}\}$$

其中，k 为预测步长，当 k 较小时为短期预测，当 k 较大是为长期预测。

传统的时间序列分析建模是针对单变量时间序列的预测主要方法。算术平均法、移动平滑法、加权移动平滑法、指数平滑法等是易于理解实现简单的基础方法。Box 和 Jenkins 提出的自回归滑动平均模型（Auto Regressive and Moving Average, ARMA）和自回归求和滑动平均（Auto Regressive Integrated Moving Average, ARIMA）模型有着广泛的应用。随后逐渐形成了完整的时间序列分析、建模和预测等理论和研究体系。

随着神经网络、人工智能等技术的发展，出现了基于卡尔曼滤波的时间序列预测、基于隐马尔科夫模型的时间序列预测、基于人工神经网络模型的时间序列预测、基于支持向量回归的时间序列预测等方法和技术，它们也广泛地应用在各种研究领域。

1.3.2.2 多变量时间序列的预测

由于多变量时间序列中通常包含一个目标时间序列和很多个变量时间序列，预测的主要任务是通过已知的多变量时间序列去持续预测目标变量。其数学描述如下：

给定一个已知的目标变量$T=\{T_1,\ T_2,\ldots\ldots,\ T_n\}$和 D 个已知的变量时间序列$X=\{X_1,\ X_2,\ldots\ldots,\ X_n\}$，其中 n 为时间序列的长度。对目标时间序列 Y 的预测是

$$X=\{X_{n+1},\ X_{n+2},\ldots\ldots,\ X_{n+k}\}\rightarrow Y'=\{Y_{n+1},\ Y_{n+2},\ldots\ldots,\ Y_{n+k}\}$$

其中，k 为持续增长的时间点。

多变量时间序列预测相比单变量时间序列的预测更加复杂和困难。所以，很多预测方法都是不同技术的结合。Wang et al(2009)提出的针对复杂时间序列的预测方法结合了 K 尺最近邻搜索、支持向量回归和差分演化（Differential Evolution）。Huang et al（2010）提出的长期时间序列预测方法结合了 K 尺最近邻法和最小二乘法支持向量机（LS-SVM）。Dhurandhar (2010)提出了利用变量序列的多步预测方法。

由于多变量时间序列中往往存在着冗余变量或者无关变量，这些都会影响预测的效果和增加预测的成本。所以，在多变量时间序列预测中，先是利用降维技术尤其是特征选择技术剔除冗余变量或者无关变量后再输入到预测模型中也很重要。陈涛（2009）[5]提出的方法利用主成分分析法对多变量时间序列进行降维处理后再采用支持向量机建立预测模型。Yuan（2011）[6]提出了基于互信息的输入变量选择的时间序列预测方法。

1.3.2.3 常用方法

当前，针对时间序列预测问题，主要的方法可以分为如下几个大类：包括基于统计学的方法、基于机器学习的方法、基于神经网络的方法。

其中，早期的时间序列预测问题研究主要集中在基于统计学的方法上。此类方法优点是具有扎实的统计学理论基础，模型可靠稳定，算法效率高。缺点是大多数方法只能处理线性序列，如常用的 ARIMA 模型。另外，一些模型的参数估计复杂，预测效果对参数敏感，难以处理前文提到的易漂变问题，所以很难在实际场景中落地应用。

典型的数理统计模型如 ARIMA 模型、Markov、Wiener 过程[9]等

表 1　常见数理统计模型

统计模型	文献	特点
ARIMA	文献[10]	对电梯门进行健康评价，拟合出健康曲线，外推得到电梯门的剩余使用寿命
马尔科夫模型	文献[11]	采用了马尔科夫理论，特点是自主预测，自主诊断，竞争学习，最终得出系统的剩余使用寿命和故障诊断定位

维纳过程	文献[12]	加入了漂移布朗运动理论
维纳随机建模	文献[13]	考虑了不同工况，不同任务场景下的寿命预测方法
PCA 与维纳过程混合模型	文献[14]	将主成分分析法与维纳过程结合，对涡轮发动机的故障预测和健康评估提出了实施方案，并取得良好的精度

数理统计模型方法是较为传统的预测评估方法，是基于统计经验的算法，要求对工程经验和历史数据有较深入的了解，而且一旦数据量过大，传统的统计模型是无法应对的。

表 2 统计模型的局限

常见线性模型	局限性
自回归滑动平均模型（Autoregressive Moving Average，ARMA）[15] 自回归积分滑动平均模型（Autoregressive Integrated Moving Average，ARIMA）[16]	1）解决问题类型单一 在线性问题上表现良好，在非线性问题上无法取得期望的结果；
	2）单一变量 对于多参数的高维数据，线性模型无法将高维数据得出有效的映射关系；
	3）噪声 线性模型受噪声，离群值等影响较大
	4）对数据要求高 线性模型要求数据的关系是独立的

非线性模型如 SVM 等神经网络模型弥补了线性模型的这些不足，在金融，交通，降水，机械等领域得到了广泛应用。

系统监测数据的增多和深度学习理论方法的火热发展使得机器学习的方法日益增多。典型的机器学习方法的应用有支持向量机（Support Vector Machine, SVM），相关向量机（Relevance Vector Machine，RVM），深度学习（deep learning，DP）等。

表 3 常见机器学习方法

机器学习方法	英文	简称
支持向量机	Support Vector Machine	SVM
相关向量机	Relevance Vector Machine	RVM

深度学习	deep learning	DP
人工神经网络	Artificial Neural Network	ANN
高斯过程	Gaussian Process,	GP

各个机器学习方法都有其独特的优势，研究者们对各个方法进行了深入的研究。表 7 是一些研究概况。

表 4　机器学习方法特点

机器学习方法	文献	特点
支持向量机	文献[16]	将支持向量机与蒙特卡洛抽样相结合的算法，对系统的疲劳裂纹进行预测，并得出寿命预测方案。
相关向量机	文献[17]	加入了特征选择方法：采用封隔器法，对涡轮发动机的工作状态进行评估，预测，是一种直接预测的方法
深度学习	文献[16-19]	
人工神经网络	文献[18]	考虑到长期退化和短期退化过程，应用于锂离子电池电量（SOH）的预测研究。
高斯过程	文献[19]	结合生存概率模型，对机械设备的退化做出评估

上世纪 90 年代以来，随着机器学习技术的兴起和发展，越来越多的研究工作专注于使用此类方法进行预测。包括使用支持向量回归法[20]、随机森林模型[21]、Adaboost 集成模型[22]等方法对序列进行预测。然而，这种简单的模型应用忽略了很多时间序列内在的本质问题，如前文所述的数据相关性问题以及窗口尺度问题等。隐马尔可夫模型（Hidden Markov Model，HMM）是一种专门针对序列问题建模的机器学习模型，然而这类方法属于线性模型，无法处理非线性数据，且模型参数估计复杂，效果也并不理想。

自 2012 年以来，深度学习的快速崛起，使得人们更多地关注起了神经网络方法。此类方法有着传统机器学习所不具备的天然优势，例如可以在线增量式的训练模型、可以自动捕捉局部特征、可以实现端到端的学习，可以对序列建模等。其中以循环神经网络为代表的方法可以专门用来处理序列问题，其已经在语音识别[23]、机器翻译[24]、视频分析[25]、文本分类[26]等诸多领域得到了成功应用，并取得了重大突破。

基于深度学习的预测是非常流行的方法，也是很多人的研究方向[27]。在工业领域，微电子的迅猛发展和进步，使得传感器变得越来与小，越来越廉价，而且灵敏度更高，

使用寿命更长，还有无线传感器的产生。依靠在设备上按照的各个位置的传感器，可以监测系统运行中的各种工况和运行数据，如温度，压力，角速度，电压电流，油量，振动等参数。在系统运行的整个周期都能 24 小时不间断的产生监测数据，为时间序列预测方法提供了足够多的数据支持。

1.3.3 深度学习在时间序列预测的应用

1.3.3.1 深度学习概念

随着计算机科学理论的发展，建立在人工神经网络基础上的深度学习（Deep Learning，DL）技术在多个领域已经取得了非常好的研究成果。深度学习的概念是由 Hinton 等于 2006 年提出来，它是基于深信度网络（Deep Brief Network, DBN）而提出的非监督贪心逐层训练算法[28]。

神经网络是模仿人体神经系统处理信息的方式来建模的，而神经系统是复杂的系统。人体有不同的脑区，分别处理不同类型的信息，如视觉、听觉、体觉、思维功能等不同的脑区，脑区的结构也有不同，因此需要利用不同类型的网络建模学习不同特征的知识。这些都被深度学习网络所借鉴。

深度神经网络（Deep Neural Networks，DNN）以神经网络为载体，重在深度，可以说是一个统称。包括了多个隐含层的循环神经网络，全连接网络，卷积神经网络。循环神经网络主要用于序列数据处理，具有一定的记忆效应，其衍生的长短期记忆网络处理长时依赖关系效果更好。卷积神经网络（Convolutional Neural Networks, CNN）则侧重空间映射，图像数据尤为适合此种网络的特征提取[29]。

1.3.3.2 LSTM 在时序数据的应用

当输入数据具有依赖性且是序列模式时，CNN 的结果一般都不太好。CNN 的前一个输入和下一个输入之间没有任何关联。

针对序列数据预测提出了循环神经网络，RNN 网络出现于 20 世纪 80 年代，最近由于网络设计的推进和图形处理单元上计算能力的提升，RNN 网络变得越来越流行。这种网络尤其是对序列数据非常有用，因为每个神经元或者单元能用它的内部存储来保存之前输入的相关信息。RNN 网络设计了不同的数量的隐含层数量，各个隐含层保存信息，选择性遗忘一些信息。这样可以提取数据的前后序列变化的数据特征。循环神经网络已经在文本处理、语音处理等领域取得了很多成果[30]。RNN 网络被应用于语音识别领域、机器翻译领域、文本生成领域、情感分析以及视频行为识别等领域。

在许多深度学习模型中，递归神经网络（Recurrent Neural Network, RNN，又名循环

神经网络），是专门处理时间序列数据的，提取数据的时序特征，因此，在处理时序数据时，RNN 比其他神经网络模型更有优势。

长短期记忆神经网络（Long Short-Term Memory，LSTM），是 RNN 的一种变体，LSTM 在神经元里加入了 3 个“门”，分别是输入门，输出门和遗忘门，有专门的记忆单元，负责记录长期和短期的数据信息，不断迭代更新记忆单元，以保持长期数据信息的有效性。

表 5　LSTM 解决的问题和方式

问题	实现方式	关键参数/结构
梯度消失	模型的训练是不断对权值 w 求导的过程，不断乘以 w，如果 w 的值小于 1，误差函数就会无限趋于 0，导致权重无法更新	权重 w
止梯度爆炸	如果 w 的值大于 1，误差函数不断乘以 w，就会无限趋于无穷，导致权重更新过快	权重 w
长期记忆	LSTM 神经元 3 个门的设计，输入负责记住一些信息，遗忘门负责丢弃一些信息	输入门，输出门，遗忘门

调研文献可知，LSMT 模型在很多领域得到应用，并取得良好的效果。

表 6　LSTM 的应用

应用领域	文献	特点
自然语言处理	文献[31]	机器翻译
自然语言处理	文献[32]	语音识别
多媒体	文献[33]	视频和音频数据分析
多媒体	文献[34]	图片标题模型搭建
生物基因	文献[35]	蛋白质序列结构
物流，交通	文献[36]	预测未来某个时间段的交通流量

然而，在可靠性与系统工程领域，LSTM 模型的实际应用很有限，特别是对于故障时间，剩余使用寿命时间序列预测的问题，相关的研究还比较少。

1.3.4 数据预处理方法

1.3.4.1 数据预处理的原因和方法

实际生产的工程数据通常由于传感器原因，人为原因，环境因素等，采集到的数据通常会有各种问题，神经网络模型对数据比较敏感，如果直接将数据输入网络，会造成模型无法充分学习到有用的特征，使得结果不准确，因此需要对数据进行预处理。

根据作者实际项目经验，数据处理相关的工作时间占据了整个项目的 70%以上。数据的质量，直接决定了模型的预测和泛化能力的好坏。它涉及很多因素，包括：准确性、完整性、一致性、时效性、可信性和解释性。而在真实数据中，我们拿到的数据可能包含了大量的缺失值，可能包含大量的噪音，也可能因为人工录入错误导致有异常点存在，非常不利于算法模型的训练。数据清洗的结果是对各种脏数据进行对应方式的处理，得到标准的、干净的、连续的数据，提供给数据统计、数据挖掘等使用。

数据可能存在的问题如表 7 所示

表 7 数据中可能存在的问题

序号	问题	描述
1	数据缺失 （Incomplete）	属性值为空的情况
2	数据噪声 （Noisy）	数据值不合常理的情况
3	数据不一致 （Inconsistent）	数据前后存在矛盾的情况
4	数据冗余 （Redundant）	数据量或者属性数目超出数据分析需要的情况
5	数据集不均衡 （Imbalance）	各个类别的数据量相差悬殊的情况
6	离群点/异常值 （Outliers）	远离数据集中其余部分的数据
7	数据重复（Duplicate）	在数据集中出现多次的数据

数据预处理的主要步骤分为：数据清理、数据集成、数据规约和数据变换。

数据清理(data cleaning) 的主要思想是通过填补缺失值、光滑噪声数据，平滑或删除离群点，并解决数据的不一致性来“清理“数据。主要方法如表 8。

表 8 数据清理方法

序号	方法	描述
1	缺失值处理	删除变量：若变量的缺失率较高（大于 80%），覆盖率较低，且重要性较低，可以直接将变量删除；定值填充；统计量填充；插值法填充；模型填充；哑变量填充
2	离群点处理	简单统计分析；3σ 原则；基于绝对离差中位数（Median

		Absolute Deviation，MAD）；基于距离；基于密度；基于聚类；
3	噪声处理	对数据进行分箱操作，等频或等宽分箱，然后用每个箱的平均数，中位数或者边界值（不同数据分布，处理方法不同）代替箱中所有的数，起到平滑数据的作用。另外一种做法是，建立该变量和预测变量的回归模型，根据回归系数和预测变量，反解出自变量的近似值。

数据集成。数据分析任务多半涉及数据集成。数据集成将多个数据源中的数据结合成、存放在一个一致的数据存储，如数据仓库中。这些源可能包括多个数据库、数据方或一般文件。

数据规约。数据归约技术可以用来得到数据集的归约表示，它小得多，但仍接近地保持原数据的完整性。 这样，在归约后的数据集上挖掘将更有效，并产生相同(或几乎相同)的分析结果。一般有维度规约和维度变换。

数据变换。主要包括规范化处理，离散化处理，稀疏化处理。

1）对数据进行规范化：数据中不同特征的量纲可能不一致，数值间的差别可能很大，不进行处理可能会影响到数据分析的结果，因此，需要对数据按照一定比例进行缩放，使之落在一个特定的区域，便于进行综合分析。特别是基于距离的挖掘方法，聚类，KNN，SVM 一定要做规范化处理；

2）离散化：数据离散化是指将连续的数据进行分段，使其变为一段段离散化的区间。分段的原则有基于等距离、等频率或优化的方法；

3）稀疏化处理：针对离散型且标称变量，无法进行有序的 LabelEncoder 时，通常考虑将变量做 0，1 哑变量的稀疏化处理，例如动物类型变量中含有猫，狗，猪，羊四个不同值，将该变量转换成 is_猪，is_猫，is_狗，is_羊四个哑变量。若是变量的不同值较多，则根据频数，将出现次数较少的值统一归为一类'rare'。稀疏化处理既有利于模型快速收敛，又能提升模型的抗噪能力。

1.3.4.2 主成分分析法

主成分分析是一种多元统计方法[37]。一般的想法是将一组相互关联的变量转换为相互独立的变量[38]。本文提出的 PCA 特征提取算法利用线性拟合思想对数据特征进行线性组合，将高维数据从原始空间映射到制定维度的低维特征空间，并且尽可能反映原始数据的特征信息[39]，数学表达式为：

$$\begin{aligned} Y_1 &= u_{11}X_1 + u_{12}X_2 + \ldots + u_{1p}X_p \\ Y_2 &= u_{21}X_1 + u_{22}X_2 + \ldots + u_{2p}X_p \\ &\cdots \\ Y_p &= u_{p1}X_1 + u_{p2}X_2 + \ldots + u_{pp}X_p \end{aligned} \tag{1.1}$$

其中，$|C-\lambda I|=0$ 为原始数据特征向量，Y_1，Y_2，…，Y_p 为线性变换后的新特征向量[40]，u_1，u_2，…，u_p 表示线性表达式的系数向量，其中，$u_i=(u_{i1}, u_{i2}, \ldots, u_{ip})$。在降维过程中，保留“主成分”[41]，“主成分”是指特征空间中方差贡献较大的数据特征方向，这样做是为了原始数据信息的损失尽量小[42]。

基于主成分分析的特征提取算法对多维数据中可能相互关联的原始特征进行线性变换，形成新的独立数据特征，并利用灰度标准差最大的新数据特征构造新的低维数据特征空间[43]；在维数提取过程中，主要数据特征信息同时减少[25]。最后，将新的数据特征输入到故障预测与评估模型中。

1.3.5 目前存在的问题

1）基于 LSTM 的时间序列预测研究应用领域较广，但是在可靠性与系统工程领域应用较少，LSTM 以其强大的时序数据处理能力，对可靠性的预测和评估会有很大的帮助。

2）LSTM 的研究多集中于时间序列数据的输入与输出的映射关系，状态 t 与状态 t+1 的转换，而没有考虑时间序列数据的空间特征，即各参量之间的影响等。Conv-LSTM 加入了卷积运算，可用于时空特征。

3）基于物理模型的评估方法需要从业人员具有较高的专业背景知识，且泛化能力不高。因此利用时序数据进行评估分析可有效解决这类问题。

1.4 软硬件平台

1.4.1 硬件平台

本文的相关模型的搭建、训练在本地单机上进行，配置如表 9：

表 9 硬件配置

配置项	参数
CPU	i5-8400
内存	8G
显卡	GTX1060
硬盘	128G SSD 1T

1.4.2 软件平台

本次实验在 Windows10 操作系统下进行。使用 Python 高级编程语言进行数据的分析、处理，模型的建立、训练与优化，Python 版本为 Python 3.6.3。Python 开发环境为在 anaconda 中安装的 Spyder。

Python 是机器学习与深度学习领域中最热门的语言。Python 之所以能够在众多语言中脱颖而出，受到机器学习与深度学习工程师的青睐，一个很重要的原因就是它有很多成熟的机器学习与深度学习开源框架与工具库，社区完善且活跃。下面介绍一些本文所用到的 Python 深度学习框架与机器学习库。

TensorFlow。TensorFlow 是深度学习领域最受欢迎的开源框架，由 Google Brain 团队开发和维护，被用于各种机器学习与深度学习算法的编程实现。TensorFlow 把神经网络的结构抽象为一张图（Graph），在一张运算图中进行张量（Tensor）各种组合计算。TensorFlow 底层是用 C/C++语言进行实现的，在上层进行了 Python 的接口封装。这样的设计使得本文 TensorFlow 框架开发的代码，既拥有 C 和 C++语言运算速度，又保证 Python 语言开发效率高的特点。本文所使用的 TensorFlow 版本为 TensorFlow 1.10.1。

Keras。Keras 被认为是简单易用的 Python 深度学习库之一。Keras 是建立在 TensorFlow、Theano 这两个框架上的更上层封装，例如基于 TensorFlow 框架 Keras 封装，隐藏了具体的张量运算、运算图定义、损失函数等细节，只提供简介的 API 供调用。Keras 文档齐全，纯 Python 编写，继承了 Python 语言的设计思想，提供一致而简洁的 API，上手快，扩展性好，极大减少了一般应用下编程的工作量。同时，Keras 提供清晰和具有实践意义的 bug 反馈。但高度的封装带来的是速度牺牲，在使用后端情况下，同一模型结构下 Keras 实现比 TensorFlow 实现训练速度慢，最多可近一倍。本文使用的 Keras 版本为 2.2.4.

Sklearn。Sklearn（Scikit-learn）是常用的机器学习工具库，基于 Numpy 和 Scipy 编写，它实现了常用的传统机器学习算法，比如支持向量机、决策树、随机森林等，还提供了很多机器学习的方法工具，比如数据标准化与归一化、特征选择、特征降维等。本次实验使用的 Sklearn 版本为 Sklearn 0.18.1.

Pandas 是一个开源的、有 BSD 开源协议的库，它为 Python 编程语言提供了高性能、易于使用的数据架构以及数据分析工具。总之，它提供了被称为 DataFrame 和 Series（对那些使用 Panel 的人来说，它们已经被弃用了）的数据抽象，通过管理索引来快速访问数据、执行分析和转换运算，甚至可以绘图（用 matplotlib 后端）。

Numpy。Numpy 是 Python 的数值计算扩展，专门用来处理矩阵，它的运算效率比列表更高效。

Matplotlib。Matplotlib 是 Python 编程语言及其数值数学扩展包 NumPy 的可视化操作界面。它为利用通用的图形用户界面工具包，如 Tkinter, wxPython, Qt 或 GTK+向应用程序嵌入式绘图提供了应用程序接口（API）。此外，Matplotlib 还有一个基于图像处理库（如开放图形库 OpenGL）的 pylab 接口，其设计与 MATLAB 非常类似--尽管并不怎么好用。SciPy 就是用 matplotlib 进行图形绘制。

1.5 模型所用数据

本文所用数据有两个，一是 NASA 数据中心提供涡轮发动机数据。这个数据集包含多工况，多传感器参数的发动机运行数据，是时间序列预测研究领域被使用比较多的数据，被很多篇论文使用。

二是某公司的锂电池的充放电实验数据。

1.6 论文主要内容与结构安排

论文的主要内容安排如下：

第一章：绪论。

主要介绍本文的研究背景，研究意义，时序数据，时序数据的预测技术，深度学习在时序数据预测的应用，数据预处理方法的国内外研究现状和目前存在的问题，研究所用的软硬件平台和所用数据。。

第二章：数据预处理。

主要有数据处理所用数据的来源，格式，数据量等的详细介绍，以 3 号发动机的数据为例，介绍主成分分析法（PCA）的原理，特征提取的步骤，去中心化和归一化的步骤，以及预处理后的数据介绍。

第三章：基于 LSTM 的 RUL 预测。

主要介绍 LSTM 神经网络的原理和结构，模型搭建的过程，分别用第二章处理过的涡轮发动机仿真数据和锂电池试验数据做预测，分别是案例一和案例二。最后用贝叶斯方法优化模型。

第四章：基于 Conv-LSTM 的 RUL 预测。

主要介绍了 LSTM 的一种变体 Conv-LSTM 的原理和结构，并用为特征提取的涡轮发动机的仿真数据作 RUL 预测。

论文的结构安排如图 3。

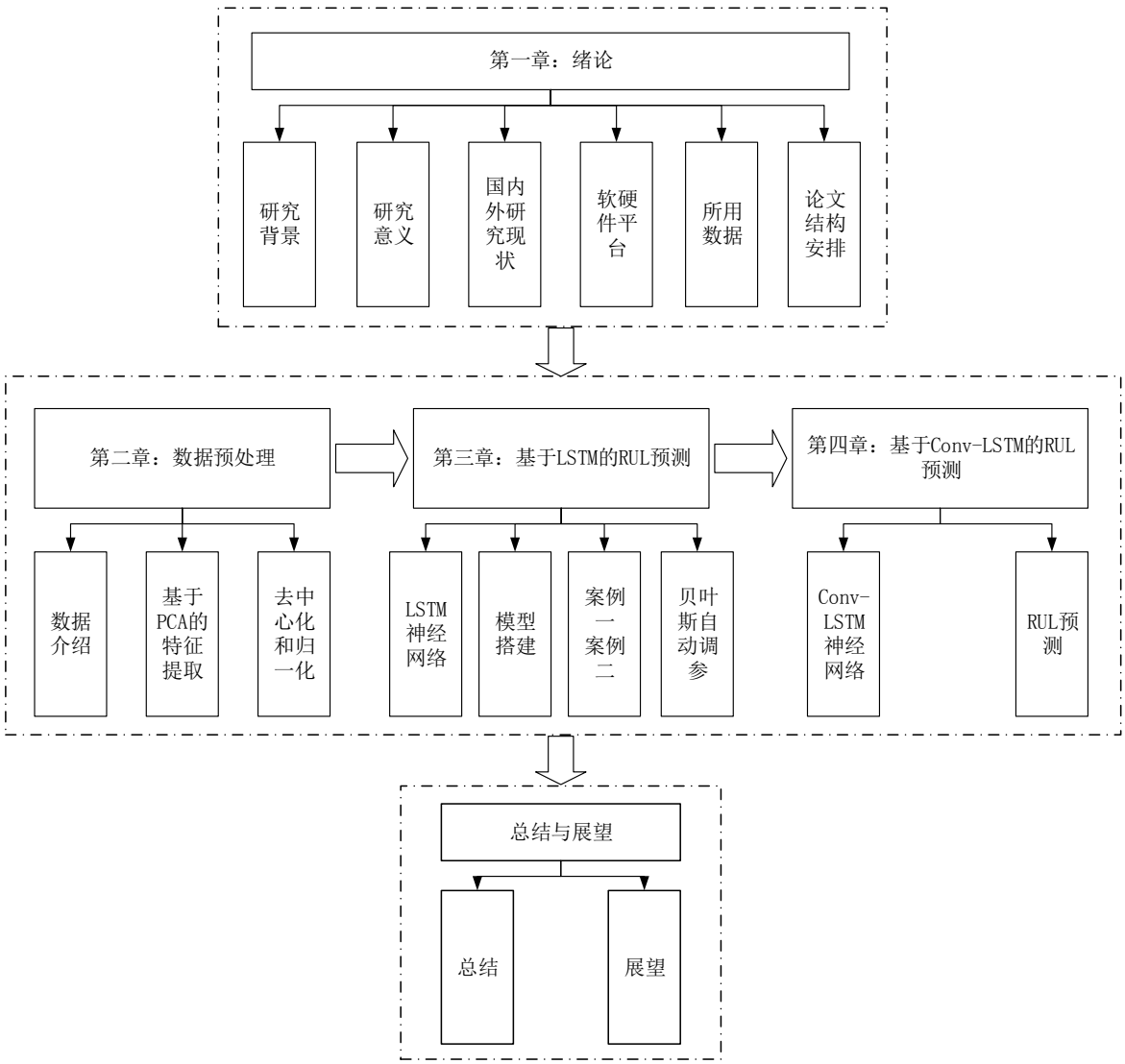


图 3　论文结构

第二章　数据预处理

实际生产的工程数据通常由于传感器原因，人为原因，环境因素等，采集到的数据通常会有各种问题，神经网络模型对数据比较敏感，如果直接将数据输入网络，会造成模型无法充分学习到有用的特征，使得结果不准确，因此需要对数据进行预处理，在本节中，对数据的预处理方法主要是基于 PCA 算法的特征提取和标准化。

2.1 数据介绍

本节所用数据是 NASA 数据中心的涡轮发动机数据。

2.1.1 NASA 发动机数据

2.1.1.1 数据来源

美国国家航空航天局（National Aeronautics and Space Administration，NASA）用软件仿真出了不同工况下的发动机运行数据，所用仿真软件是 C-MAPSS，该软件由 NASA 研发，用于模拟大型商用涡轮发动机的运行数据。代码的实现是在 MATLAB 与 Simulink 环境中完成，程序给出了多个 API（Application Programming Interface，应用程序接口），用户可根据实际情况调整参数，如传感器噪声，运行状态，故障模式，环境条件，温度，湿度，压力，训练轨迹等。图 4 是模拟发动机的基本片段。

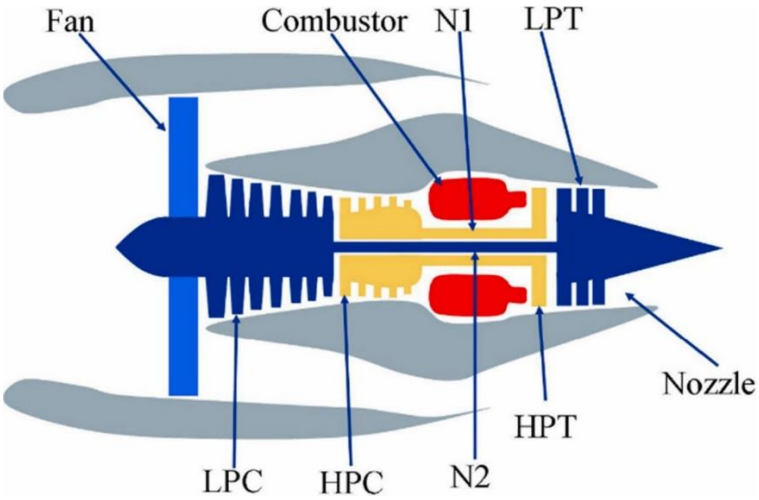


图 4　模拟发动机的基本片段

2.1.2 数据详细信息

发动机的数据文件格式为 txt 文件，分为训练集 train.txt 和测试集 test.txt，本文所用数据文件为 train.txt，共包含 218 台发动机的数据，共约 5 万条记录。以 3 号发动机的数据为例，数据是 26 维，前 2 个参量分别是发动机的编号，寿命周期，后面 24 维数据是 3 个工况参数，后面是 21 个运行参数，这 24 维数据需要作数据预处理。共有 150 个

循环数。部分数据如表 10。

表 10　3 号发动机的部分参量数据（前 10 行）

发动机编号	循环（cycle）	工况 1	工况 2	工况 3	运行参数 1	…	运行参数 21
3	1	19.9998	0.7013	0.0	491.19	…	14.6936
3	2	20.0071	0.7012	0.0	491.19	…	14.7728
3	3	25.0071	0.6213	80.0	462.54	…	8.6033
3	4	19.9982	0.7	0.0	491.19	…	14.6214
3	5	0.0003	0.0	100.0	518.67	…	23.2717
3	6	35.0044	0.84	60.0	449.44	…	8.9173
…	…	…	…	…	…	…	…
3	148	42.0011	0.84	40.0	445.0	…	6.4113
3	149	25.0031	0.6213	80.0	462.54	…	8.6195
3	150	20.0017	0.7	0.0	491.19	…	14.7289

表 10 中，参量寿命周期可作为剩余使用寿命取值的参考，单个发动机总的寿命周期数与当前寿命周期的差值作为当前的剩余使用寿命 RUL，如表 11 所示。

表 11　剩余使用寿命的取值依据

寿命周期	剩余使用寿命 RUL
1	150
2	149
3	148
4	147
5	146
6	145
…	…
148	3
149	2
150	1

218 台发动机的剩余使用寿命在区间[150，225]上。

表 11 展示了 3 号发动机的部分数据，由于维度较高，因此只展示了部分参量数据，

数据可视化如图 5。

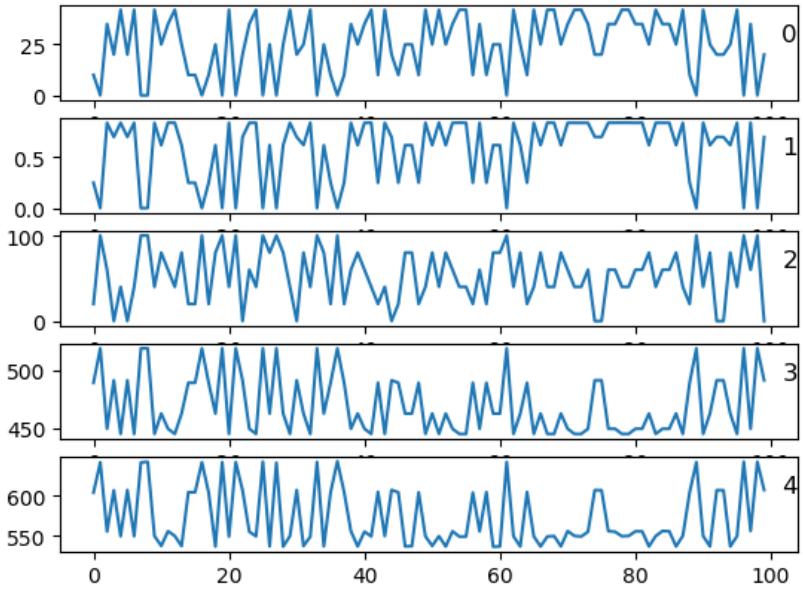


图 5 原始数据可视化

图 5 是可视化后的数据，从图中可看出各参量的变化趋势及数量级，对之后的数据预处理提供了直接，感官上的依据。

2.2 基于 PCA 的特征提取

2.2.1 PCA 原理

实际生产的工程数据通常由于传感器原因，人为原因，环境因素等，采集到的数据通常会有各种问题，神经网络模型对数据比较敏感，如果直接将数据输入网络，会造成模型无法充分学习到有用的特征，使得结果不准确，因此需要对数据进行预处理，在本文中，对数据的预处理方法主要是基于 PCA 算法的特征提取和标准化

主成分分析是一种多元统计方法[44]。一般的想法是将一组相互关联的变量转换为相互独立的变量[45]。本文提出的 PCA 特征提取算法利用线性拟合思想对数据特征进行线性组合，将高维数据从原始空间映射到制定维度的低维特征空间，并且尽可能反映原始数据的特征信息[20]，数学表达式为：

$$\begin{aligned} Y_1 &= u_{11}X_1 + u_{12}X_2 + \ldots + u_{1p}X_p \\ Y_2 &= u_{21}X_1 + u_{22}X_2 + \ldots + u_{2p}X_p \\ &\cdots \\ Y_p &= u_{p1}X_1 + u_{p2}X_2 + \ldots + u_{pp}X_p \end{aligned} \tag{2.1}$$

其中，$|C-\lambda I|=0$ 为原始数据特征向量，Y_1，Y_2，…，Y_p 为线性变换后的新特征向量

[21]，u_1，u_2,…，u_p表示线性表达式的系数向量，其中，$u_i=\left(u_{i1}，u_{i2},…，u_{ip}\right)$。在降维过程中，保留“主成分”[46]，“主成分”是指特征空间中方差贡献较大的数据特征方向，这样做是为了原始数据信息的损失尽量小[47]。

基于主成分分析的特征提取算法对多维数据中可能相互关联的原始特征进行线性变换，形成新的独立数据特征，并利用灰度标准差最大的新数据特征构造新的低维数据特征空间[48]；在维数提取过程中，主要数据特征信息同时减少[49]。最后，将新的数据特征输入到故障预测与评估模型中。

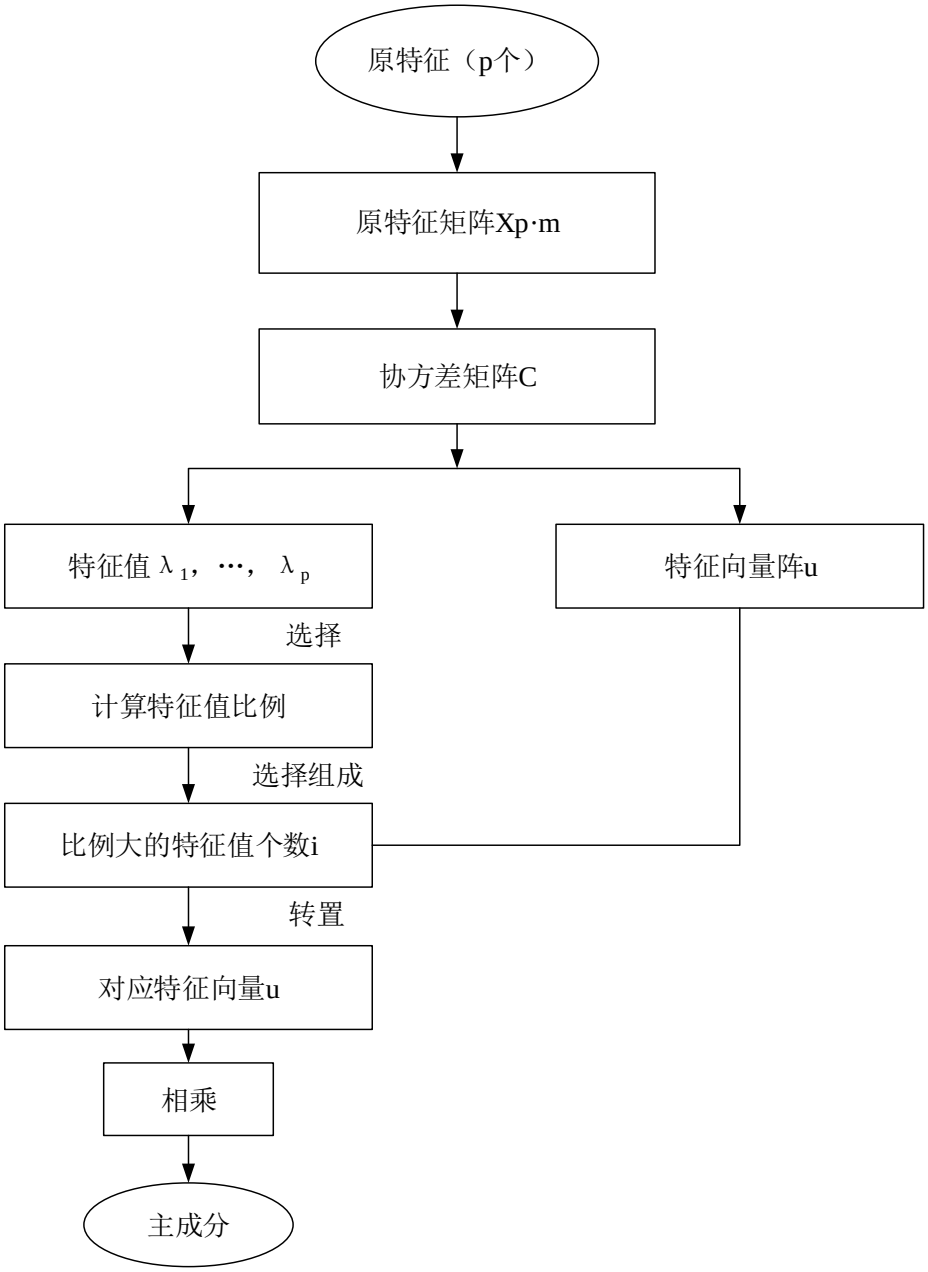


图 6　基于 PCA 的特征提取算法

1）由原特征矩阵 X，得出协方差矩阵 C[50]

公式中的原特征矩阵$u_1,\ u_2,\ldots,\ u_p$由 p 维特征向量$X_1,\ X_2,\ldots,\ X_p$组成，协方差矩阵 C 为：

$$C=\begin{bmatrix} cov(X_1,\ X_1) & cov(X_1,\ X_2) & \ldots & cov(X_1,\ X_p) \\ cov(X_2,\ X_1) & cov(X_2,\ X_2) & \ldots & cov(X_2,\ X_p) \\ \ldots & & & \\ cov(X_p,\ X_1) & cov(X_p,\ X_2) & \ldots & cov(X_p,\ X_p) \end{bmatrix} \tag{2.2}$$

式中，$\mathrm{X}=\left(X_1,\ X_2,\ldots,\ X_p\right)^T$，C 为对称矩阵。

2）由协方差矩阵 C 得出特征值$\lambda_1,\ \lambda_2,\ldots,\ \lambda_p$，以及标准正交特征向量$u_1,\ u_2,\ldots,\ u_p$。

$$\left|\mathrm{C}-\lambda\mathrm{I}\right|=0 \tag{2.3}$$

可得出协方差 C 的特征值$\lambda_i\left(i=1,2,\ldots,\ \mathrm{p}\right)$，根据特征多项式

$$\left(C-\lambda I\right)u=0 \tag{2.4}$$

计算出 p 维特征向量矩阵

$$u_{p\times p}=\left(u_1,\ u_2,\ldots,\ u_p\right) \tag{2.5}$$

3）主要成分的选择[51]。本文中，选取主成分维数的依据是协方差矩阵 C 的特征值比例大小顺序，标准正交化之后的特征向量矩阵表示为 u，由大到小排列，特征值个数取较大的 i 作为新特征的维数。

$$\lambda_1,\ \lambda_2,\ldots,\ \lambda_i\ \left(i<p\right) \tag{2.6}$$

给出相对应的主成分维数的特征向量矩阵

$$u_{p\times i}=\left(u_1,\ u_2,\ldots,\ u_i\right) \tag{2.7}$$

4）求主成分，即新特征。以上步骤完成后，将降维后的标准正交特征向量矩阵 u 转置，得到转置矩阵u^T，乘以原矩阵 X，得到u^TX，由此得出新综合特征变量[52]，此时便得出了主成分Y的转置矩阵Y^T：

$$Y^T=\begin{bmatrix} T_1^T \\ T_2^T \\ \ldots \\ T_i^T \end{bmatrix}=u^TX=\begin{bmatrix} u_1^T \\ u_2^T \\ \ldots \\ u_i^T \end{bmatrix}\bullet X\begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1p} \\ u_{21} & u_{22} & \ldots & u_{2p} \\ \ldots & \ldots & \ldots & \ldots \\ u_{i1} & u_{i2} & \ldots & u_{ip} \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \\ \ldots \\ X_p \end{bmatrix} \tag{2.8}$$

通过上述变换过程，标准正交特征向量矩阵 u 中的元素是方程（2.1）中每个特征向量的系数，系数绝对值的大小也决定了原始特征量在新特征量中的比例[53]。本文中，选

取主成分的根据是，数据原始特征协方差矩阵 C 的特征值，新数据特征的维数是取特征值比较大的前几个特征值的$i(i<p)$，包括原始数据的大部分信息。

数据的主成分特征提取出来，降维完成，最后将降维生成的新数据输入到神经网络模型，进行故障预测与健康评估等。

2.2.2 PCA 特征提取步骤

设有 m 条 n 维数据。则 PCA 进行特征提取的步骤如表 12 所示。

表 12　PCA 特征提取步骤

序号	运算
1	将原始数据按列组成 n 行 m 列矩阵 X；
2	将 X 的每一行进行零均值化，即减去这一行的均值；
3	求出协方差矩阵$C=\frac{1}{\mathrm{m}}XX^{T}$；
4	求出协方差矩阵的特征值及对应的特征向量；
5	将特征向量按对应特征值大小从上到下按行排列成矩阵，取前 k 行组成矩阵 P；
6	$Y=PX$ 即为降维到 k 维后的数据。

在本文中，对 218 台发动机的数据都进行特征提取，由于数据量比较大，以 3 号发动机的数据为例，介绍特征提取的过程。

1）3 号发动机有 150 条 24 维的数据，即 m=150，n=24，将数据组成 24 行 150 列的矩阵 X，如表 13 所示。

表 13　3 号发动机的矩阵 X

	1	2	3	4	…	150
1	19.9998	20.0071	25.0071	19.9982	…	25.004
2	0.7013	0.7012	0.6213	0.7	…	0.6214
3	0.0	0.0	80.0	0.0	…	80.0
4	491.19	491.19	462.54	491.19	…	462.54
5	607.76	607.88	536.61	607.72	…	537.44
6	1486.02	1479.31	1258.63	1486.79	…	1273.77
…	…	…	…	…	…	…
22	100.0	100.0	84.93	100.0	…	84.93
23	24.51	24.45	14.08	24.47	…	14.01

24	14.6936	14.7728	8.6033	14.6214	…	8.5062

2）将 X 的每一行进行零均值化，即减去这一行的均值；

求出每一列的均值：

$$M = mean(A) \tag{2.9}$$

即：

$$M(m_{11},\cdots,\ m_{1,24}) = \begin{matrix} (a_{11}+a_{12}+\cdots+a_{150})/150 \\ \cdots \\ (a_{12}+a_{22}+\cdots+a_{150})/150 \end{matrix} \tag{2.10}$$

矩阵 X 的每一列减去均值，得到新矩阵如表 14。

表 14 X 零均值化的矩阵

	1	2	3	4	…	150
1	-4.1567	-4.1494	0.8506	-4.1583	…	0.8475
2	0.1249	0.1248	0.0449	0.1236	…	0.045
3	-49.7333	-49.7333	30.2667	-49.7333	…	30.2667
4	18.3212	18.3212	-10.3288	18.3212	…	-10.3288
5	28.7942	28.9142	-42.3558	28.7542	…	-41.5258
6	67.9463	61.2363	-159.4437	68.7163	…	-144.3037
…	…	…	…	…	…	…
22	2.6121	2.6121	-12.4579	2.6121	…	-12.4579
23	3.9908	3.9308	-6.4392	3.9508	…	-6.5092
24	2.373	2.4522	-3.7173	2.3008	…	-3.8144

3）求出协方差矩阵 $C = \frac{1}{\mathrm{m}} XX^T$；

得出的协方差矩阵 C 为 24*24 的矩阵，如表 15。

表 15 协方差矩阵 C（24*24）

	1	2	3	4	…	24
1	0.1148	0.0007	1.3787	-2.9505	…	0.0138
2	0.0007	0.0	0.0249	0.0397	…	-0.0011
3	1.3787	0.0249	16.4893	6.1396	…	0.4643
4	-2.9505	0.0397	6.1396	13.985	…	-0.7541
5	2.0911	-0.0744	1.9681	-9.8611	…	0.9583
6	-9.8606	0.6111	23.0276	53.5038	…	-10.7887

…	…	…	…	…	…	…
22	-0.2464	0.0271	-0.5178	0.2818	…	-0.3949
23	0.4677	-0.0113	-0.2564	-3.3834	…	0.2458
24	0.0138	-0.0011	0.4643	-0.7541	…	0.0909

4）求出协方差矩阵的特征值及特征向量；

特征值为表 16。

表 16 特征值

序号	特征值
1	2308.8
2	593.224
3	-366.897
4	59.9844
5	40.2161
6	27.8632
…	…
22	-0.0010739
23	-0.000182736
24	1.20955e-06

特征矩阵为表 17。

表 17 特征矩阵

	1	2	3	4	…	24
1	-0.00482043	-0.01843	0.030271	0.00380091	…	-0.000368562
2	0.000296299	0.000781158	-0.00177037	-0.00816049	…	-0.0248167
3	-0.0153224	0.072949	-0.00519488	-0.361062	…	0.000251516
4	0.0381042	0.146772	-0.0711496	0.00149787	…	0.000978037
5	-0.0232165	-0.0528585	-0.0521184	0.0275404	…	-0.000156722
6	0.297755	0.25063	0.287151	-0.278644	…	-0.000131738
…	…	…	…	…	…	…
22	0.0086499	-0.011729	0.0246994	0.00335604	…	0.00362842
23	-0.00604225	-0.0291807	-0.0061106	-0.0327823	…	-0.0079101
24	-0.00916325	-0.00781377	0.0260425	0.0311644	…	-0.00411856

5）将特征向量按对应特征值大小从上到下按行排列成矩阵，取前 k 行组成矩阵 P；

根据特征值排序结果可得，最大的前 10 个值及对应的行号如表 18。

表 18 最大的 10 个特征值及行号

行号	特征值
1	2308.8
2	593.224
4	59.9844
5	40.2161
6	27.8632
10	5.40508
11	3.12634
12	1.65999
16	0.0891466
17	0.0305255

特征值大小理解为所含信息，数据降维肯定会损失信息，所以在纬度一定的情况下我们需要尽可能的保留多一点信息。根据特征值的大小，另 k=7，选取较大的 7 个特征值所对应的特征向量组成矩阵 P，如表 19 所示。

表 19 特征矩阵 P

	1	2	3	4	…	24
1	-0.00482043	-0.01843	0.030271	0.00380091	…	-0.000368562
2	0.000296299	0.000781158	-0.00177037	-0.00816049	…	-0.0248167
3	-0.0153224	0.072949	-0.00519488	-0.361062	…	0.000251516
4	0.0381042	0.146772	-0.0711496	0.00149787	…	0.000978037
5	-0.0232165	-0.0528585	-0.0521184	0.0275404	…	-0.000156722
6	0.297755	0.25063	0.287151	-0.278644	…	-0.000131738
7	-0.107043	-0.246258	-0.206933	-0.0621249	…	3.86443e-05

6）$Y = PX$ 即为降维到 k=7 维后的数据，其中，X 为本文需要降维的 3 号发动机数据的原始矩阵。

表 20 降维后的矩阵 Y（7*150）

	1	2	3	4	…	150
1	-5986.05	-5984.98	-5331.6	-5991	…	-5321.18
2	-2574.91	-2572.19	-2619.34	-2572.05	…	-2613.88
3	969.781	968.631	737.956	967.898	…	741.773

4	1496.63	1492.81	1388.86	1496.72	…	1392.16
5	2247.99	2240.54	2023.35	2248.65	…	2036.83
6	28.8552	29.0936	164.381	29.1601	…	159.93
7	2135.56	2133.85	1988.82	2136.17	…	1988.12

将矩阵 Y 转置，可得 3 号发动机降维后的数据，如表 21。

表 21　降维后的新数据

	Var1	Var2	Var3	Var4	Var5	Var6	Var7
1	-5986.05	-2574.91	969.781	1496.63	2247.99	28.8552	2135.56
2	-5986.05	-2574.91	969.781	1496.63	2247.99	28.8552	2135.56
3	-5986.05	-2574.91	969.781	1496.63	2247.99	28.8552	2135.56
4	-5986.05	-2574.91	969.781	1496.63	2247.99	28.8552	2135.56
5	-5986.05	-2574.91	969.781	1496.63	2247.99	28.8552	2135.56
6	-5986.05	-2574.91	969.781	1496.63	2247.99	28.8552	2135.56
…	…	…	…	…	…	…	
148	-5680.59	-2650.48	924.476	1316.96	2133.28	125.972	2158.83
149	-5680.59	-2650.48	924.476	1316.96	2133.28	125.972	2158.83
150	-5680.59	-2650.48	924.476	1316.96	2133.28	125.972	2158.83

可用 Python 语言实现上述需求。

2.3 去中心化和归一化

特征提取之后，由于各个字段的数据量级有差异，神经网络对数据非常敏感，对于小量级的数据若是非常重要，在神经网络中会被降低重要度，因此还需对数据进行去中心化和归一化处理。

去中心化的步骤为：

1）求每一列的平均值 μ 。

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.11}$$

2）每一列都减去平均值 μ 。

$$x_i = x_i - \mu \tag{2.12}$$

归一化的方式是线性 Max-Min 化。归一化的公式为：

$$x_i' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{2.9}$$

式中，x_{min} 为每一列的最小值，x_{max} 为每一列的最大值，x 为原数据值，x_i' 为归一化后的值。

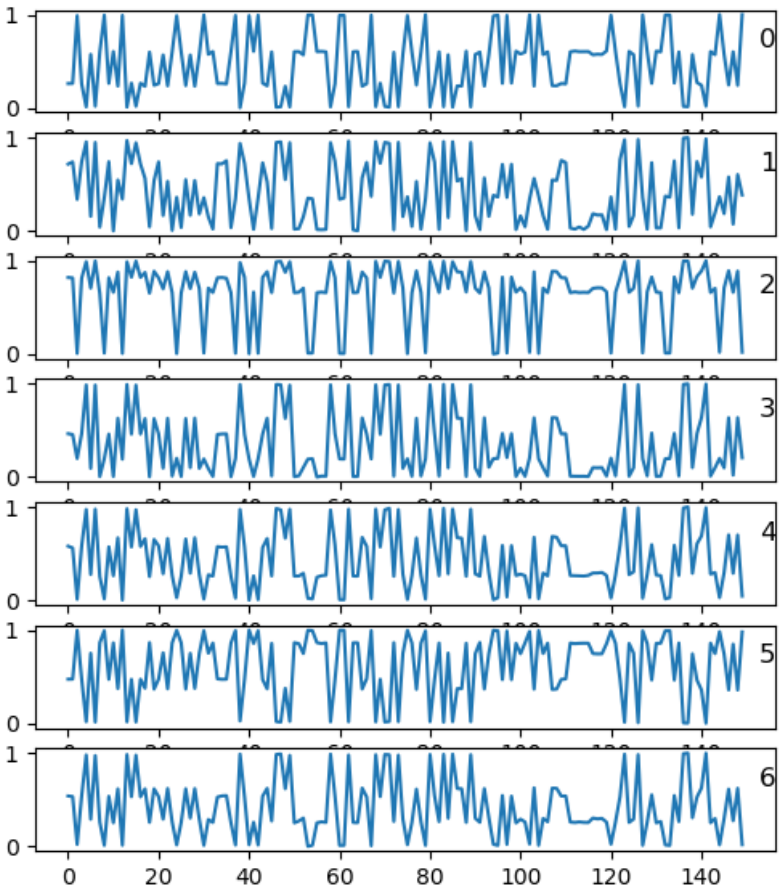


图 7 降维归一化的数据可视化图

图 7 是 3 号发动机降维归一化后的数据，可见数据之间相关性较弱，已经提取出了原始数据的主要特征。为第三章的基于 LSTM 的发动机寿命预测做好了数据准备。

2.4 本章小结

利用一种 PCA 方法对数据作了特征提取，并对数据作了去中心化和归一化处理。

针对高维的监测数据，为了方便后续的模型训练预测，提高效率和准确度，利用 PCA 算法对数据进行降维处理，提取出数据的主要特征，与此同时，通过去中心化和归一化处理，使数据更容易被神经网络处理。为基于 LSTM 的时间序列预测做好数据准备。

第三章　基于 LSTM 的 RUL 预测

对于高维的海量的传感器数据，经过第二章的数据预处理之后，利用 LSTM 得到传感器数据和剩余使用寿命 RUL 的映射，搭建预测模型，用第二章的数据和某锂电池数据做成两个案例。

3.1 LSTM 神经网络

3.1.1 网络层

神经网络模型的组成通常有一个输入层 X，任意数量的隐藏层，输出层 ŷ，每层之间的一组权重和偏差，w 和 b 等组成

每个隐藏层的激活函数 σ 的选择。Keras 库封装的 LSTM 层默认的激活函数是 tanh。图 9 显示了 2 层神经网络的架构（在计算神经网络中的层数时，通常会排除输入层）。

神经网络模型通常有一个输入层 X，任意数量（通常由经验决定）的隐藏层，一层输出层 ŷ，层与层之间存在权重和偏差，通常权重用小写字母 w 表示，偏差用 b 表示。

每层隐含层都有激活函数，是上一层的输出与下一层的输入的映射关系，主要作用是增加模型的非线性。默认是线性激活函数，即 $f\left(x\right)=x$。在本文中，我们将使用 Adam 激活函数。图 8 是 2 层神经网络的结构(通常输入层不计入神经网络模型的层数)。

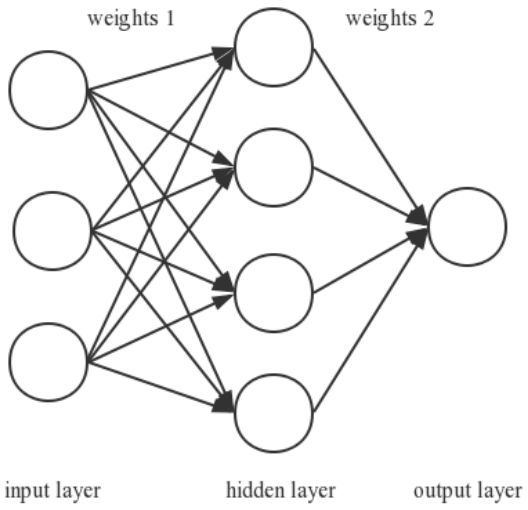


图 8　2 层神经网络的体系结构

简单的 2 层神经网络的输出是

$$\mathrm{y}=\sigma\left(\mathrm{W}_2\sigma\left(W_1x+b_1\right)+b_2\right) \tag{3.1}$$

在公式（3.1）中，影响输出 y 的 2 个超参数是权重 w 和偏差 b。权重和偏差的正确值决定了预测和分类的强度。从输入数据后，权重和偏差微调的过程被称为训练神经网

络。

训练过程的每次迭代都包含以下步骤：

1）计算预测输出 ŷ，称为前馈

2）更新权重和偏差，称为反向传播

下面的顺序图说明了该过程。

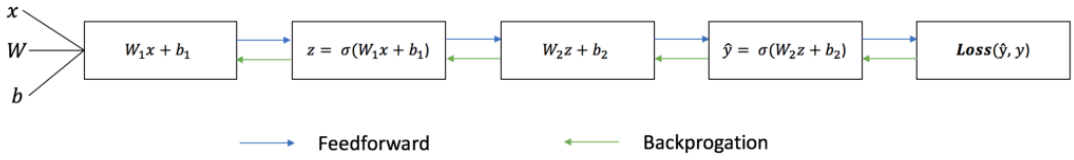


图 9　神经网络迭代过程

其中，前馈（Feedforward）的运算是简单的微积分，对于简单基本的 2 层神经网络，神经网络的输出是：

$$y=\sigma\left(W_2\sigma\left(W_1x+b_1\right)+b_2\right) \tag{3.2}$$

3.1.2 LSTM 神经元结构

LSTM 是 RNN 比较重要的模型，在自然语言处理（NLP），语音识别，以及时间序列相关的领域等都有广泛的应用。如果把 LSTM 当成黑盒子看待，可以分为以下关键变量：

1）输入：h_{t-1}（t-1 时刻的隐藏层）和 x_t（t 时刻的特征向量）

2）输出：h_t（加 softmax 即可作为真正输出，否则作为隐藏层）

3）主线（记忆）：c_{t-1} 和 c_t

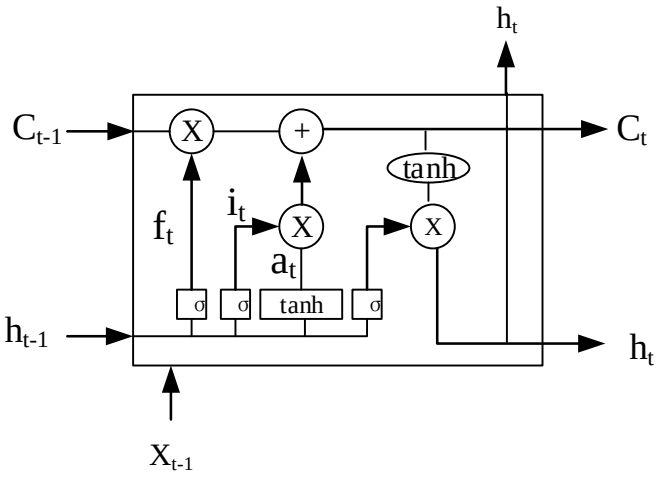


图 10　LSTM 神经元结构

h_{t-1}和x_t联合起来控制了三个门，并且是输入的唯一来源，所以划分为输入部分；h_t与真正的输出只隔了一层 softmax（图 10 中没画出），是输出的直接来源，所以划分为输出部分。当然它同时又是下一个 LSTM cell 的输入（图 10 中往右跑的h_t），但是在当前 cell，它仅与输出相关。

C_{t-1}和C_t始终与外界隔离开来，显然是作为 LSTM 记忆或者主线剧情的存在。主线进来后，首先受到遗忘门的衰减作用，接着输入门控制“补给大小”给主线补充能量生成全新的主线。这一衰一补的过程完成了主线的更新。接着在输出门的控制下生成新的输出h_t。

3.1.3 LSTM 公式

3.1.2 给出了 LSTM 的模型结构，在本小节中介绍 LSTM 公式。

从图 10 可以看出，在每个序列索引位置 t 时刻向前传播的除了和 RNN 一样的隐藏状态$h(t)$，还多了另一个隐藏状态，如图中上面的长横线。这个隐藏状态我们一般称为细胞状态(Cell State)，记为$C(t)$。

除了细胞状态，LSTM 图 10 中还有了很多的结构，这些结构一般称之为门控结构(Gate)。LSTM 在在每个序列索引位置 t 的门一般包括遗忘门，输入门和输出门三种。下面研究图中 LSTM 的遗忘门，输入门和输出门以及细胞状态。

遗忘门（forget gate），从名字可以看出是控制是否遗忘的，在 LSTM 中即以一定的概率控制是否遗忘上一层的隐藏细胞状态。

输入的有上一序列的隐藏状态$h(t-1)$和本序列数据$x(t)$，通过一个激活函数，一般是$sigmoid$，得到遗忘门的输出$f(t)$，由于$sigmoid$的输出$f(t)$在[0,1]之间，因此这里的输出$f(t)$代表了遗忘上一层隐藏细胞状态的概率。用数学表达式即为：

$$f(t)=\sigma\left(W_f h_{(t-1)}+U_f x(t)+b_f\right) \tag{3.3}$$

其中W_f，U_f，b_f为线性关系的权重和偏置，与 RNN 类似，σ为$sigmoid$激活函数。

输入门（input gate）负责处理当前序列位置的输入。隐藏状态$h(t)$的更新由两部分组成，第一部分是$o(t)$，它由上一序列的隐藏状态$h(t-1)$和本序列数据$x(t)$，以及激活函

数 $sigmoid$ 得到，第二部分由隐藏状态 $C(t)$ 和激活函数 $tanh$ 组成,

$$\begin{aligned} i(t) &= \sigma\left(W_i h(t-1) + U_i x(t) + b_i\right) \\ a(t) &= tanh\left(W_a h(t-1) + U_a x(t) + b_a\right) \end{aligned} \tag{3.4}$$

其中 W_i，U_i，b_i，W_a，U_a，b_a 为线性关系的系数和偏倚，和 RNN 中的类似。σ 为 $sigmoid$ 激活函数。

LSTM 的细胞更新。前面的遗忘门和输入门的结果都会作用于细胞状态 $C(t)$。细胞状态 $C(t)$ 由两部分组成，第一部分是 $C(t-1)$ 和遗忘门输出 (t) 的乘积，第二部分是输入门的 $i(t)$ 和 $a(t)$ 的乘积，即：

$$C(t) = C(t-1) \odot f(t) + i(t) \odot a(t) \tag{3.5}$$

其中，$\odot$ 为 Hadamard 积。

有了新的隐藏细胞状态 $C(t)$，接下来是输出门（output gate）。隐藏状态 $h(t)$ 的更新由两部分组成，第一部分是 $o(t)$，它由上一序列的隐藏状态 $h(t-1)$ 和本序列数据 $x(t)$，以及激活函数 $sigmoid$ 得到，第二部分由隐藏状态 $C(t)$ 和 tanh 激活函数组成，即：

$$\begin{aligned} o(t) &= \sigma\left(W_o h(t-1) + U_o x(t) + b_o\right) \\ h(t) &= o(t) \odot tanh\left(C(t)\right) \end{aligned} \tag{3.6}$$

3.2 模型搭建

本节主要研究内容是在深度学习框架 Keras 的帮助下，设定网络形状、激活函数、损失函数，加入正则项、偏置项、约束项和 Dropout 等[54]。

1）搭建过程

Keras 库中封装了两种类型的神经网络模型，分别是层层叠加的序贯模型(Sequential)和层与层之间函数连接的函数式模型(Model)，函数式模型的应用更多，更为广泛[55]，序贯模型是函数式模型的一个特例。本论文中使用的顺序模型是该功能模型的最短版本，是最简单的从第一到最后的无分支顺序的线性模型，是由多个网络层组成的线性堆栈。

正则化项和 Dropout 都是为了增强网络的泛化能力[55]，在一定程度上避免过拟合，在这里仅做简要介绍。正则化通过在损失函数中加入权重相关的参数，使得模型权重的更新受到约束，从而降低模型对训练集数据的“逐点拟合”现象[56]。正则项会导致训练

误差增大[57]，在本文研究中，正则项没有带来好的结果。Dropout 的运作方式简单来说就是随机丢掉网络层中一部分节点，仅使用剩下的节点进行计算，每次迭代中丢掉哪些节点都是随机的。当设定的 Dropout 比例很高时，例如 0.8，此时相当于训练了很多个网络，而多个相同结构的模型结果的组合几乎总会比单独模型的结果好一点。

优化器优先使 SGD、ADAM、RMSprop，后两种优化器的参数略有不同，其中最为重要的参数是学习率（learning rate）和衰减（decay）。学习率控制梯度下降的速度，过小的学习率会造成训练速度过慢，训练时间过长，而过大的学习率可能导致模型无法收敛，在最优解附近震荡徘徊。因此衰减（decay）的加入可以使刚开始训练时使用较大的学习率，随着训练进行，逐渐减小学习率来避免在最优解附近徘徊的情况。实际使用时需要根据训练的迭代次数调整衰减快慢。学习率还可以通过调用底层函数来自由调整。图 11 展示了初始学习率为 0.001，decay=0.000005 时的学习率变化曲线。

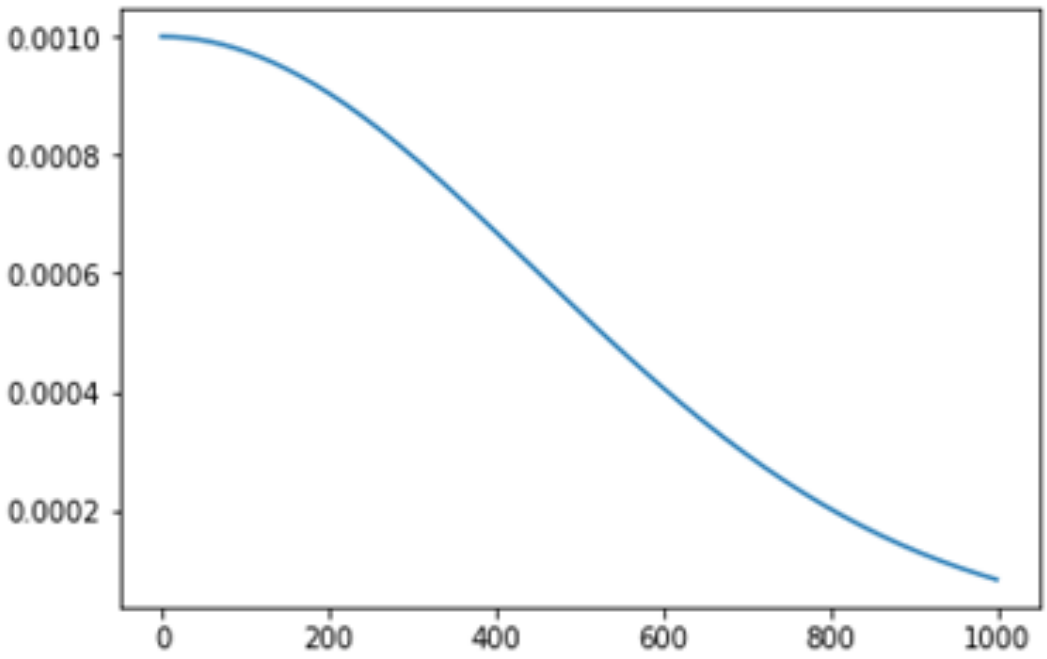


图 11　学习率在训练过程中的衰减曲线

常见的优化器有基于随机梯度下降（SGD），自适应优化器（AdaGrad），RMSProp 优化器，Adam 优化器等[56]。

表 22　常用优化器及其特点

名称	特点
SGD	SGD 的基本小批量 SGD 优化算法在深度学习中取得了良好的效果。然而，还有几个问题需要解决。选择一个合适的初始学习速度是困难的。学习速率调整策略受预先确定的调整规则的限制。

	对每个参数采用相同的学习速度，是一个非常非凸误差函数优化过程。如何避免大量的局部最优解和对点的入侵[57]。
Adarad	AdaGrad 优化器具有具有特定的参数学习率，它根据训练期间模型更新参数的频率可以进行自适应的调整。参数接收到的迭代更新越多，更新就越小。
RMSProp	克服 AdaGrad 梯度了快速下降的问题，在许多应用中表现出良好的学习适应性。特别是，非平稳目标函数比基本 SGD、Mementum 和 AdaGrad 更能提高性能。
Adam	通过将更新后的步长限制在一个粗略的范围内(初始学习率)，可以自然地实现逐步退火过程。

随机梯度下降优化器(SGD)[59]。基于概率梯度下降法(SGD)的优化算法在许多研究和工程领域都非常重要。许多理论或工程问题可以转换成最小化目标函数的数学问题。据宇恩达教授说，gradientodient 是指从山上跑到山谷的最底点，以最快(最急)的速度向最低位置急进的人。

自适应(AdaGrad)[60]。针对简单的 SGD 和 Momentum 问题，2011 年 John Duchi 和其他人推出了 AdaGrad 优化算法(Adaptive Gradient)。小的步骤会被更新，而稀疏参数会被大的步骤更新。

RMSProp 优化器[61]。RMSProp 是在教学计划中 Geoffrey Hinton 教授提到的算法，它通过组合梯度指数移动平均来调整学习效率。非稳定目的函数的情况下，可以很好地收敛。

Adam 优化器[62]。2014 年 12 月，Kingma 和 Lei Ba 两位学者提出了 AdaGrad 和 RMSProp 优化算法两者优点的组合。同时考虑梯度的一次矩估计(梯度的平均)和梯度的二次矩估计(梯度的非集中方差)，计算更新步长。

2）模型结构可视化

简单的神经网络就是用网络层叠加而成，每层可调整的参数有很多。究竟什么样的网络适合什么样的问题没有直接的对应关系，全靠不断地试错。深度学习领域近年来也出现很多自动构建网络的方法，例如文献中的自动构建 RNN，但都需要强大的算力才能实现。本文的神经网络均由手动构建。

建立一个 4 层的神经网络，前三层由 LSTM 节点构成，每层节点数目是根据经验暂时设为 48、36、24 个 LSTM 节点，第四层由一个 Dense 节点负责综合输出，网络结构

如图 12。之所以选择四层而不是五层、六层或更高层，是因为经过试验，四层以上的 LSTM 无法收敛。

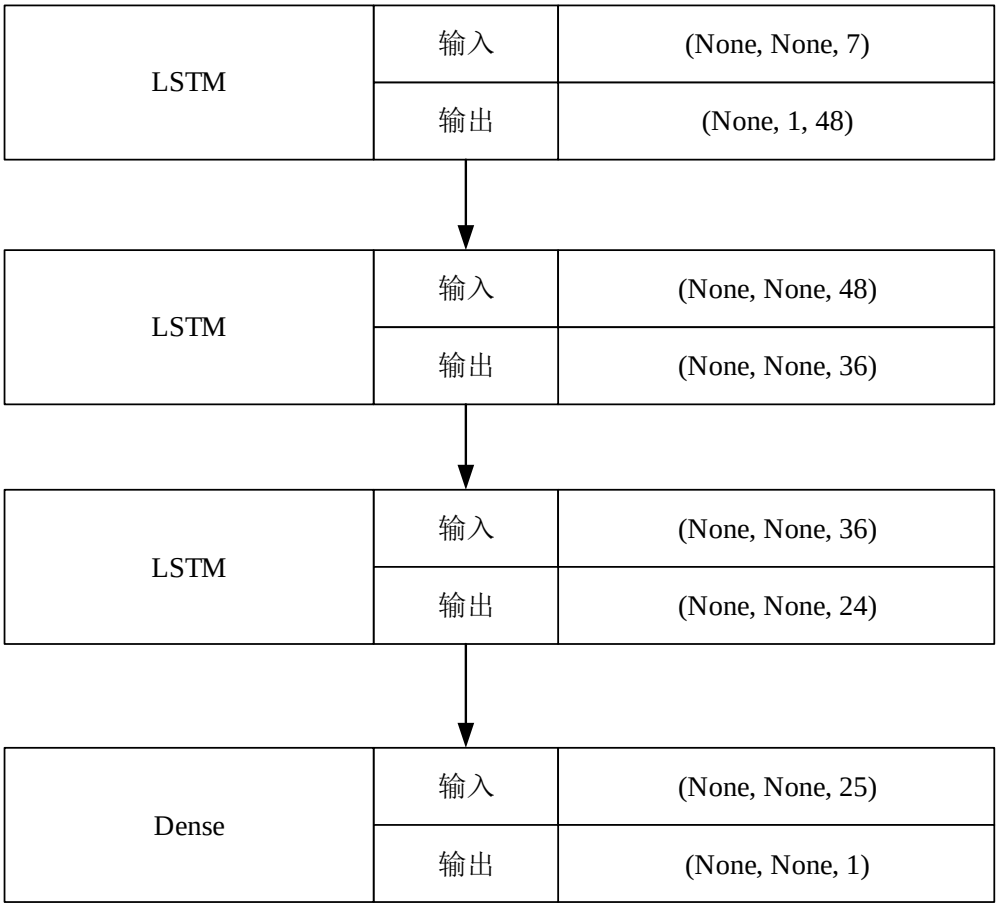


图 12　模型结构

该神经网络总参数个数为 32137，模型的参数越多，模型的能力就越强，模型覆盖面就越广，就越容易将问题的解包括进去。参数过少的模型覆盖的解空间就越小，实际表现为模型欠拟合、不收敛。32137 个参数对于 45918 条数据来说，已经足够，通常网络参数个数不要超过训练样本数。相比图像识别、自然语言处理应用中动辄上百的单层节点数，数十数百万的模型参数，本文故障预测所用的神经网络要小得多，一方面也说明数据量较小。

图 12 是案例一的模型，在案例二中，模型层数不变，会根据训练结果适当调整模型的超参数。

3.3 案例一

经过上述神经网络搭建过程可知，神经网络中可以调整的参数众多，要获得理想的结果需要反复试错。现阶段采用的 stateless LSTM 就是普通的 LSTM，在此展示集中不同参数组合的试验结果。

3.3.1 数据

所用数据为第二章中经过预处理过的涡轮发动机仿真数据。用前 180 台发动机数据训练，后 38 台发动机数据预测。

3.3.2 模型一

将该人工网络在训练集上训练 epoch=50 遍，batch 大小为单台发动机的数据记录的个数，优化器 SGD，学习率为 0.1，训练过程的误差变化见图 13，图中横轴代表训练轮次 epoch，纵轴代表误差 MSE 和 MAE。

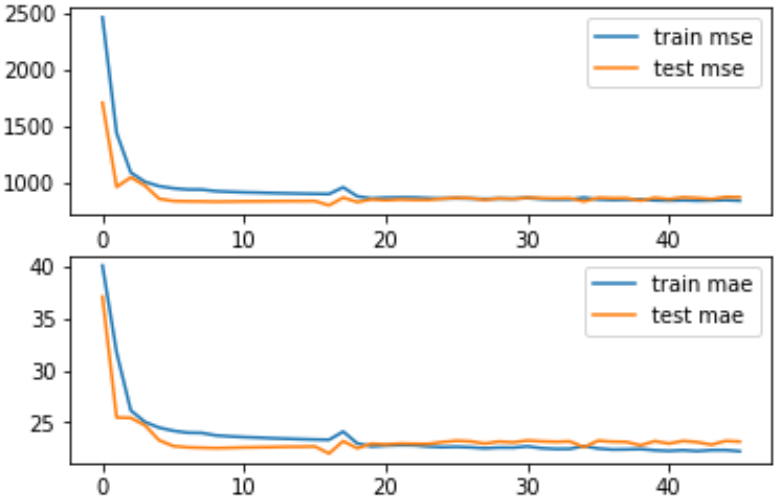
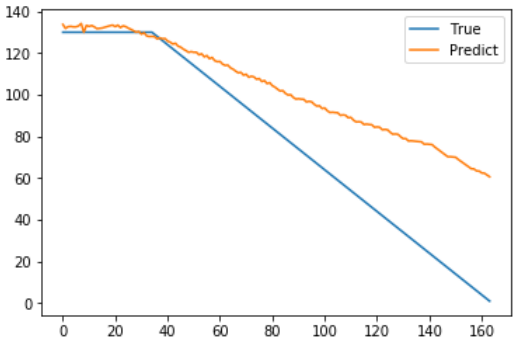
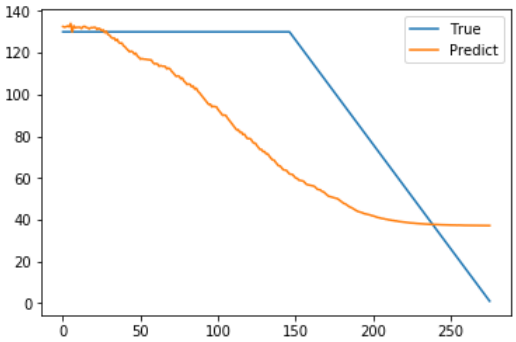


图 13　LSTM epoch loss 变化图

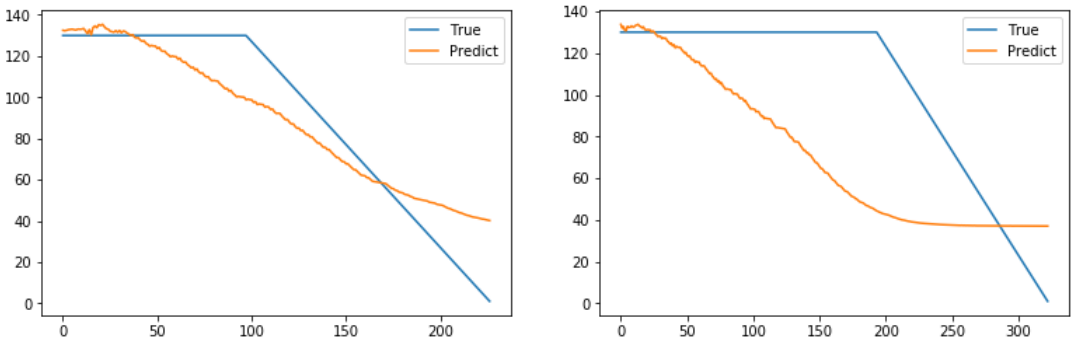
将上述模型在编号为#2，#8，#111 和#147 样本上进行预测的结果如图 14。



（a）#2



（b）#8



（c）#111　　（d）#147

图 14　LSTM 预测结果（部分样本）

如图 14 所示，搭建的 LSTM 已经可以数据中获取一定的寿命趋势。但是与真实寿命数据误差比较大，说明神经网络模型没有很好的收敛，查找资料，分析原因一是学习率（learning rate）过大，模型在最优解附近震荡无法收敛，在训练轮次之内，误差保持一个较大的值，且不再下降。基于以上分析结果，减小学习率，调整后的参数和结果在模型二中。

3.3.3 模型二

将该人工网络在训练集上训练 epoch=100 遍，batch 大小为单台发动机的数据记录的总个数，优化器选用 SGD，学习率为 0.001，训练过程的误差变化见图 15，图中横轴代表训练轮次 epoch，纵轴代表均方误差误差 MSE 和平均绝对误差 MAE。

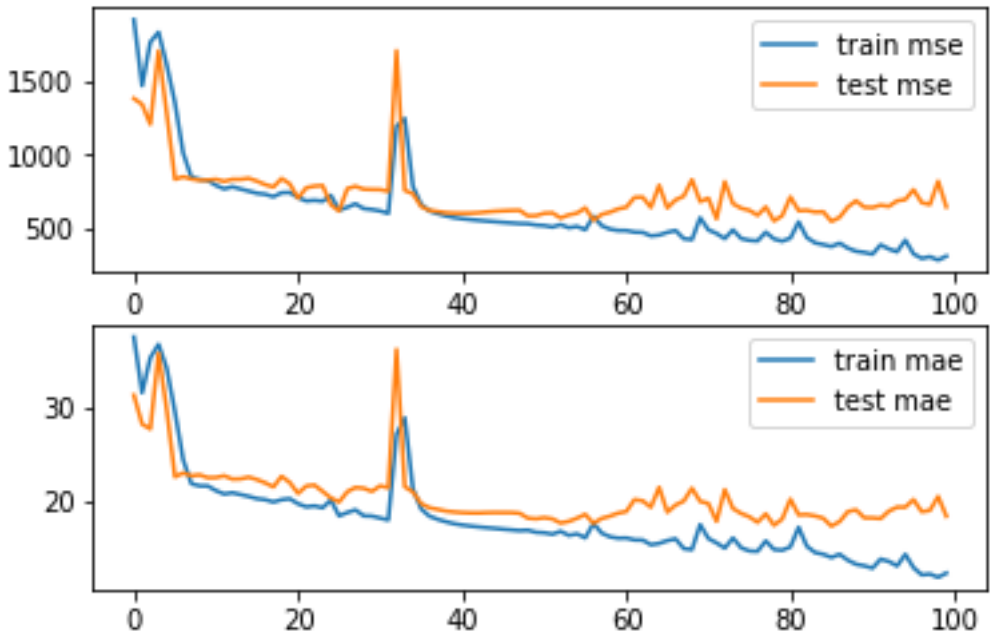
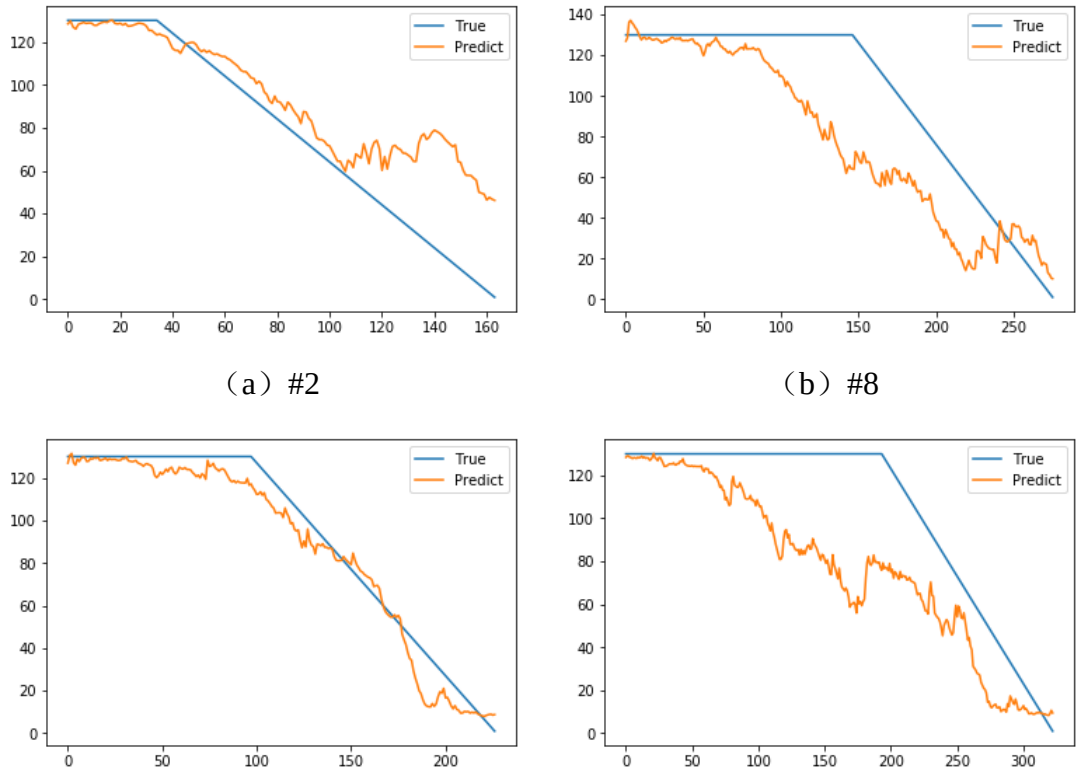


图 15　LSTM epoch loss 变化图

将上述模型在编号为#2，#8，#111 和#147 样本上进行预测的结果如图 16。



（a）#2　　（b）#8

（c）#111　　（d）#147

图 16　LSTM 预测结果（部分样本）

如图16所示，在模型一基础上修改的模型二已经可以数据中获取更优的寿命趋势。但是与真实寿命数据误差还是比较大。在后续的试错中，增加 dropout 削弱了模型的能力，但是对结果没有任何改进，加入正则项的效果与 dropout 类似。这两个都是用于防止过拟合，对于模型二中的问题的解决没有有效的帮助。关于优化器，很多论文里都会用 SGD。SGD 虽然能达到极小值，但是比其它算法用的时间长，而且可能会被困在局部极值点，因此可将优化器改为当前比较流行的 Adam 优化器。

3.3.4 模型三

将该人工网络在训练集上训练 epoch=100 遍，批次大小（batch size）大小为单台发动机的运行数据记录的总个数，在模型二的基础上修改优化器，选用 Adam，学习率为 0.001。

训练过程的误差变化见图 17，图中横轴代表 epoch，纵轴代表误差 MSE 和 MAE。

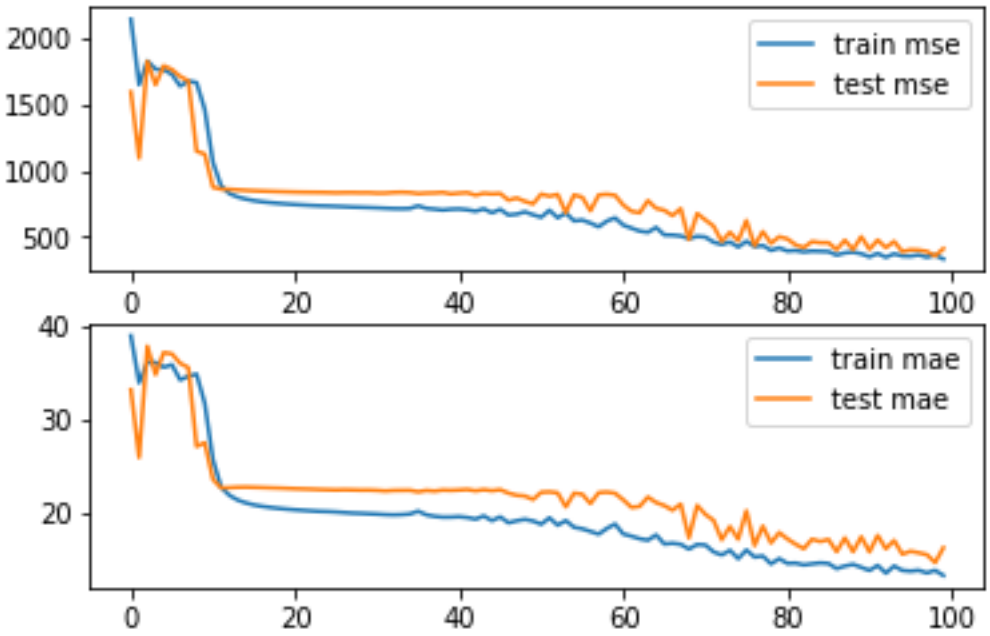
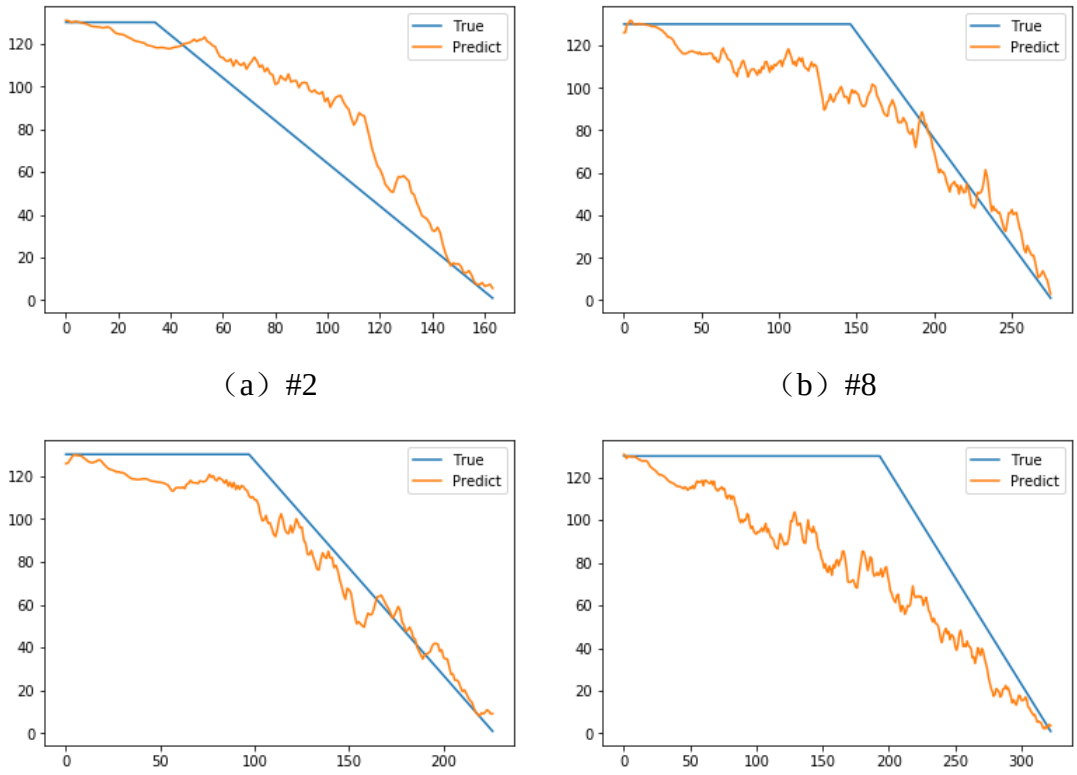


图 17　LSTM epoch loss 变化图

将上述模型在编号为#2，#8，#111 和#147 样本上进行预测的结果如图 18。



（a）#2　　（b）#8

（c）#111　　（d）#147

图 18　LSTM 预测结果（部分样本）

如图 18 所示，搭建的 LSTM 已经可以从新数据集中获取相对较好的寿命趋势。在训练集上的平均绝对误差（mae）为 13.34，在测试集上的 mae 为 16.30，说明特征提取对模型预测结果的提升有帮助。由图 18 可知，模型在测试集的 8 号和 11 号发动机

拟合效果比较好，验证集输出的结果上传到 NASA 的数据预测中心，得到的分数（Score）为 1995.66 分，排进前 20 名，是一个相对较好的得分

3.4 案例二

3.4.1 数据

为了验证模型的普适性和在实际项目数据中的表现，在某电池数据上进行验证，数据由某公司提供的锂电池实验数据。

数据由实验得出，共约 30G 的数据量。数据为 MATLAB 的数据格式，以.mat 为后缀，用 MATLAB 打开后，可以看到总共有 50 节电池的数据，每节电池都有 100 个左右的充电周期（cycle）及对应的 100 个左右的放电周期，每个周期大约有 50000 条记录，是多维的数据，包括温度，湿度，时间等字段。

以 5 号电池数据为例，这是从原始数据中提取出来的特征，共有截取了前 7 行，9 维的数据，输入前 8 列分别是 t-1 时刻充放电次数，充电时间，放电时间，充放电时间，使用时间，温度，容量，对应第九列是 t 时刻的容量。用前 40 节电池训练，后 10 节电池预测。

表 23　5 号锂电池部分数据

充放电次数 (t-1)	充电时间 (t-1)	放电时间 (t-1)	充放电时间(t-1)	使用时间 (t-1)	放电深度 (电压) (t-1)	温度 (t-1)	容量 (t-1)	容量 (t)
1	10802.32	2660.313	13462.64	13462.64	3.676931 4	12.46348 1	0.739935 1	1.166544
2	10804.68	2826.0	13630.68	27093.32	3.720909 3	13.22908 7	1.166544	1.105252 1
3	10803.51	2818.578	13622.09	40715.422	3.746790 7	12.76795	1.105252 1	1.096030 2
4	10803.28	2713.812	13517.09	54232.51	3.559910 1	14.48548 4	1.096030 2	1.075445 1
5	10802.07	2786.578	13588.656	67821.172	3.757978 1	12.21069 5	1.075445 1	1.065549 3
…	…	…	…	…	…	…	…	…
103	10803.43	2530.172	13333.60	1094626.436	3.739355 2	12.64653 9	0.156836 9	0.144906 4

按照第二章中的数据预处理方法对锂电池数据进行去中心化和归一化，由于数据本身维度并不高，而且各个参量的相关性较弱，因此原始数据已经代表了数据的主要特征，因此没有进行特征提取。得到的新数据如表

表 24　预处理后的数据

充放电次数(t-1)	充电时间 (t-1)	放电时间 (t-1)	充放电时间(t-1)	使用时间 (t-1)	放电深度 (电压) (t-1)	温度 (t-1)	容量 (t-1)	容量 (t)
0	0.3226	0.3226	0	0	0.7843	0.9362	0.3882	0.4973
0.0098	0.9677	0.9831	0.0079	0.0046	0.6723	0.5488	0.4973	0.625
0.0196	1	0.9677	0.0176	0.0112	0.9572	0.6527	0.625	0.6184
0.0294	0.6452	0.5829	0.0269	0.0242	0.9733	0.5323	0.6184	0.6053
0.0392	0.6774	0.9573	0.0401	0.0383	0.9572	0.8642	0.6053	0.6193
…	…	…	…	…	…	…	…	…
1	0	0.4829	1	1	0.8462	0.3452	0.0826	0.0811

3.4.2 预测过程与结果

模型结构如图 19。

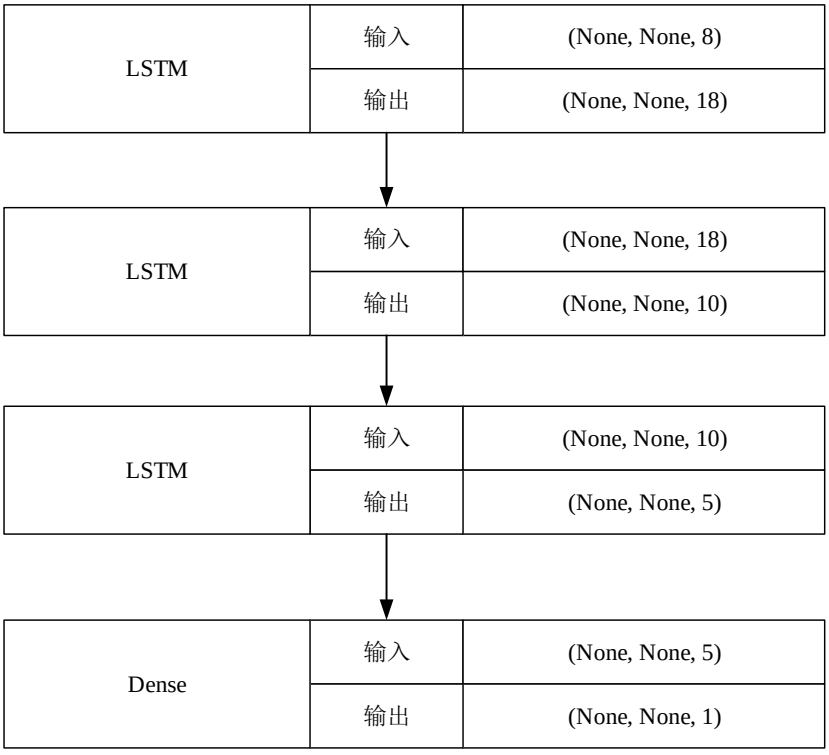


图 19　LSTM 模型结构

训练过程如下：

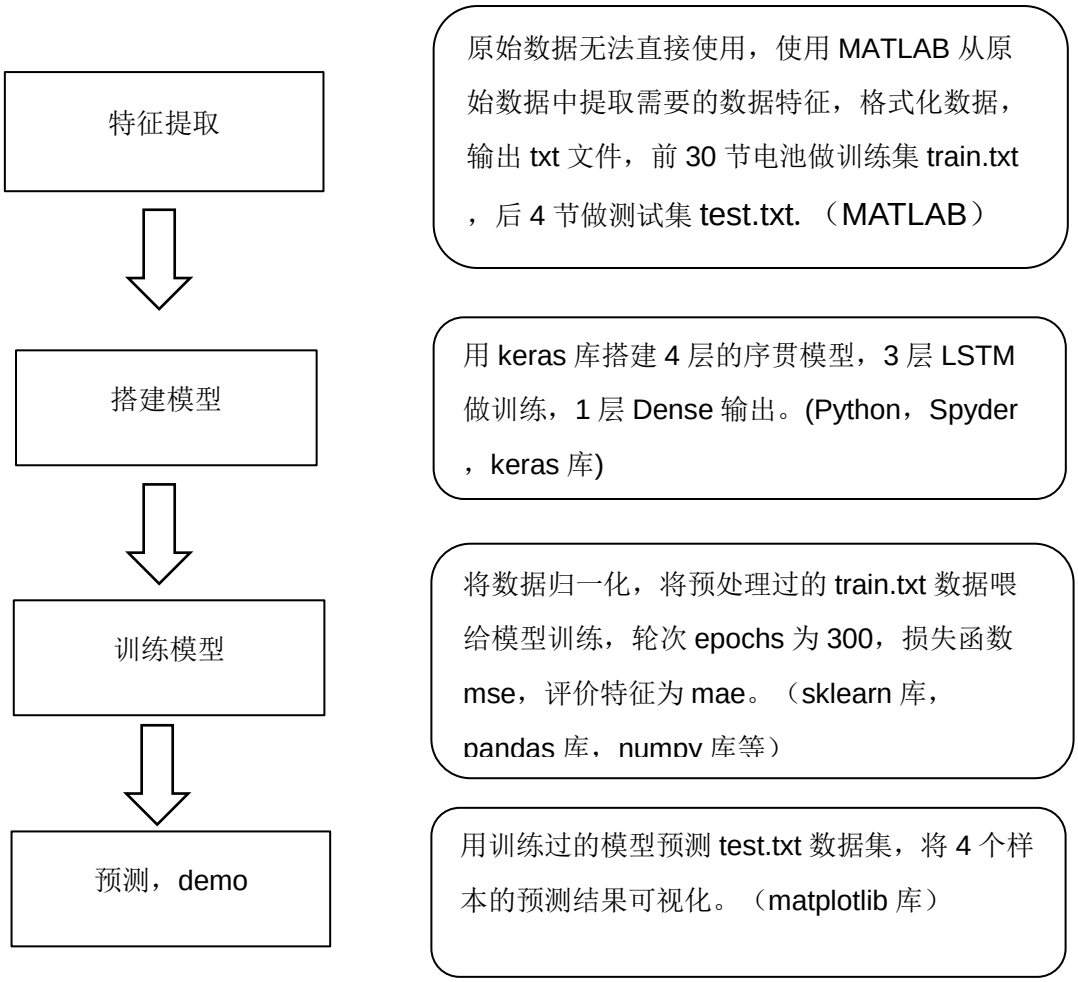

特征提取
原始数据无法直接使用，使用 MATLAB 从原始数据中提取需要的数据特征，格式化数据，输出 txt 文件，前 30 节电池做训练集 train.txt，后 4 节做测试集 test.txt.（MATLAB）
搭建模型
用 keras 库搭建 4 层的序贯模型，3 层 LSTM 做训练，1 层 Dense 输出。(Python，Spyder，keras 库)
训练模型
将数据归一化，将预处理过的 train.txt 数据喂给模型训练，轮次 epochs 为 300，损失函数 mse，评价特征为 mae。（sklearn 库，pandas 库，numpy 库等）
预测，demo
用训练过的模型预测 test.txt 数据集，将 4 个样本的预测结果可视化。（matplotlib 库）

图 20　训练过程

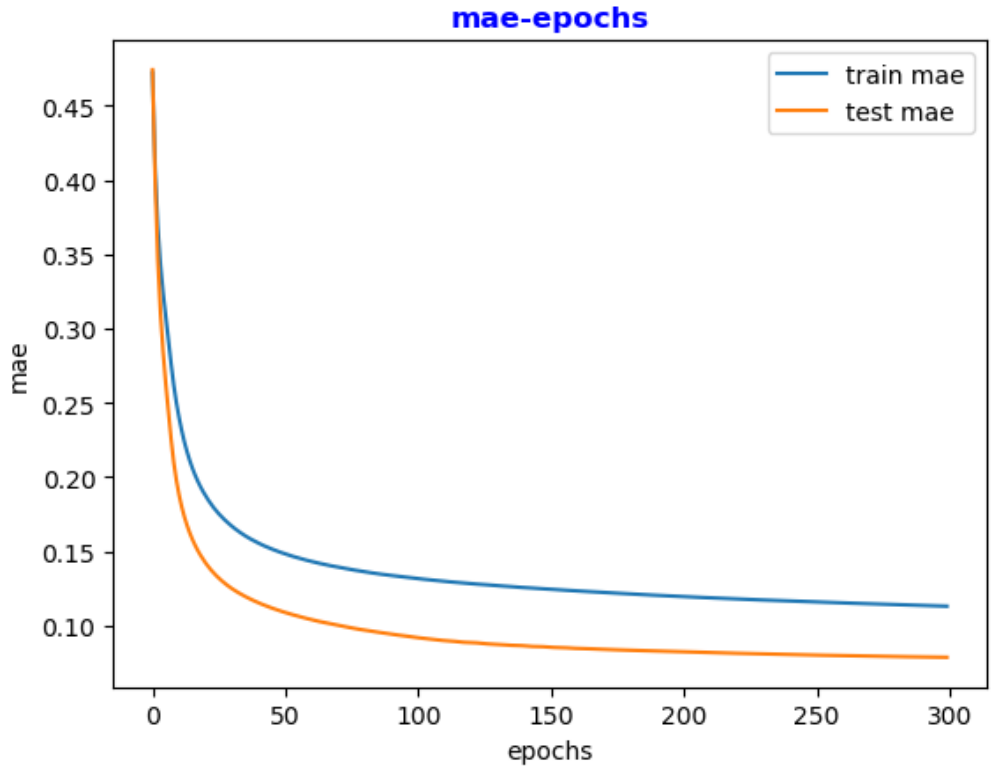
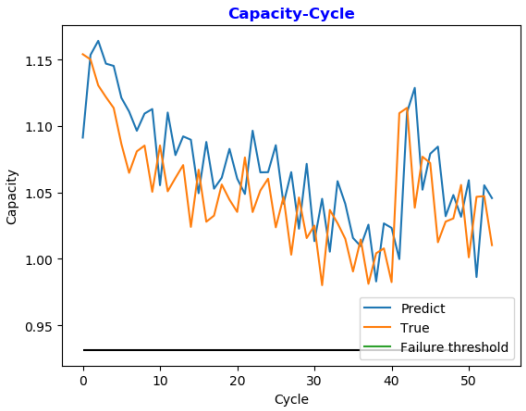


图 21　训练过程中误差变化

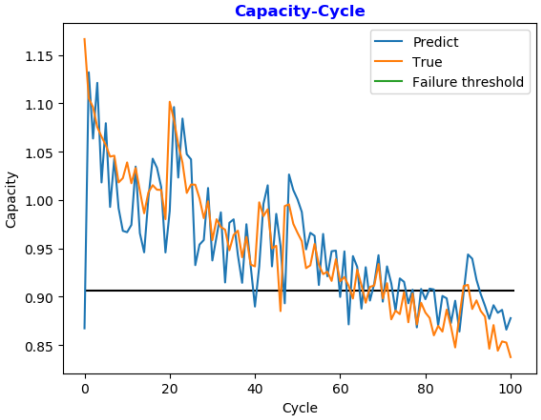
这是训练过程中的误差变化，因为学习率在收敛过程中会自适应减小，以便更好的趋近最优解。

表 25　不同轮次的误差比较

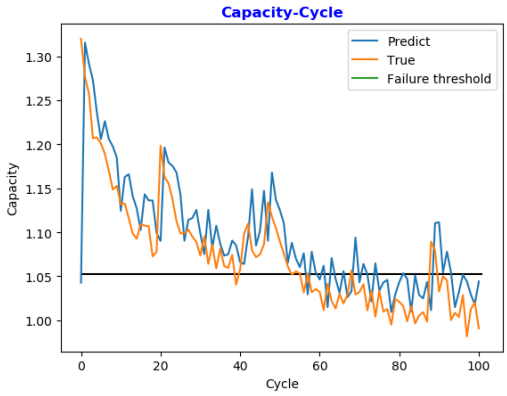
训练轮次	训练误差(MAE)	测试误差(MAE)
30	0.17	0.12
300	0.11	0.08

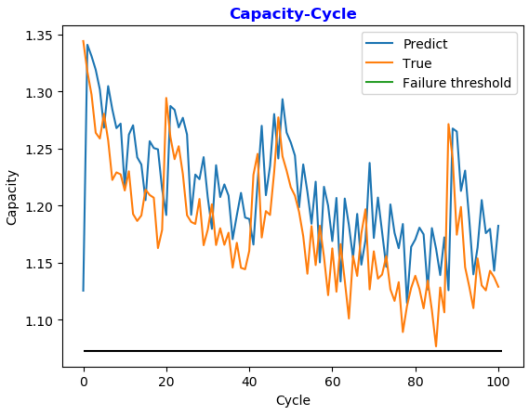


（a）#53



（b）#54





（c）#55　　（d）#56

图 22　模型在 4 个样本的预测结果

这是测试集 4 个样本的预测结果，黄线是真实值，蓝线是预测值，黑色横线是失效阈值，大小为起始容量的 80%。

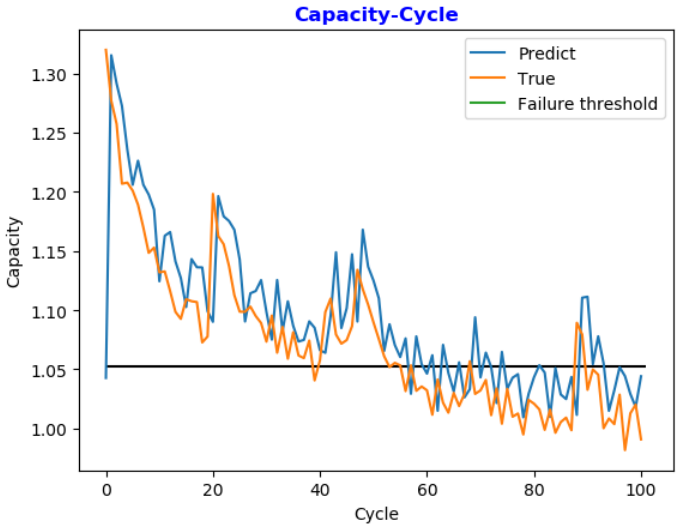


图 23　55 号电池预测结果

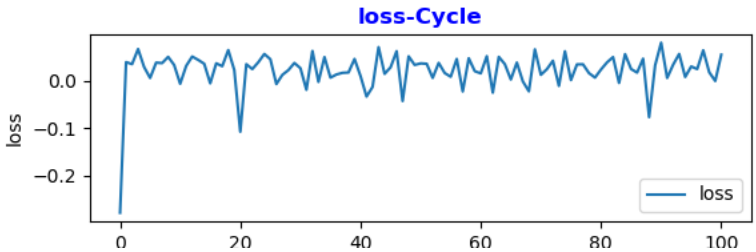


图 24　55 号电池预测误差变化

这是 55 号电池的预测结果，模型已经可以从测试数据中学习到容量变化的趋势，拟合效果较好，真实值在 53 次充放电循环达到失效阈值，预测值在 56 次达到失效阈值

第一点预测误差较大，因为第一个循环前面没有输入，所以模型自己给了一个值，

预测是在第二个循环开始

图 24 是预测值和真实值的差值，误差基本在[-0.05,0.05]区间内。

误差在第 20，87 个点较大，是因为真实容量值出现较大的震荡，模型在处理这种震荡时会出现较大的误差。

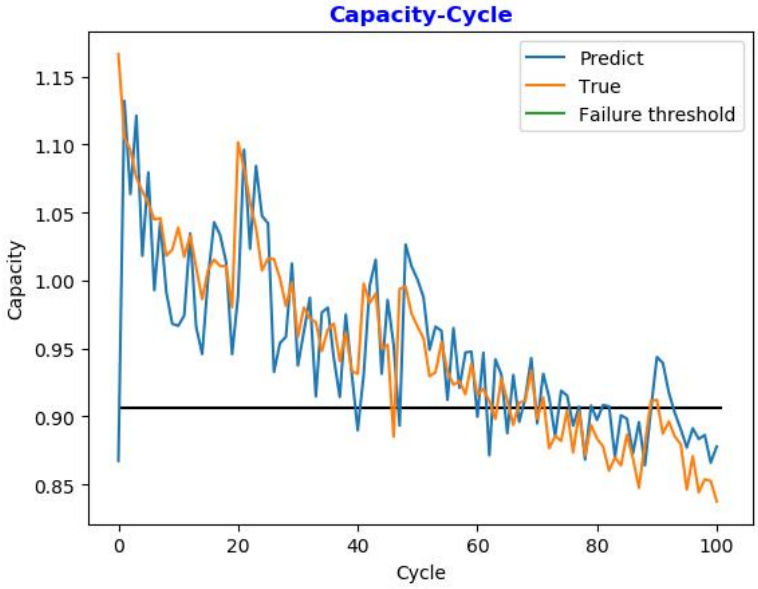


图 25　54 号电池预测结果

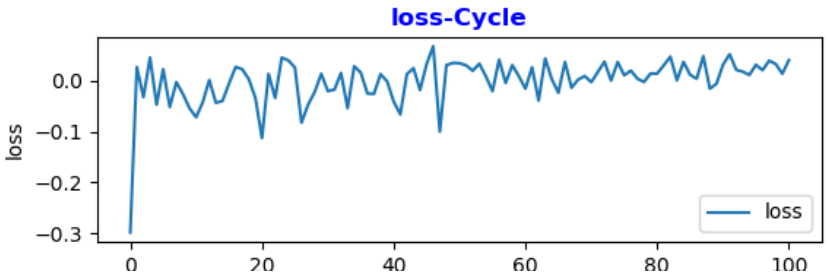
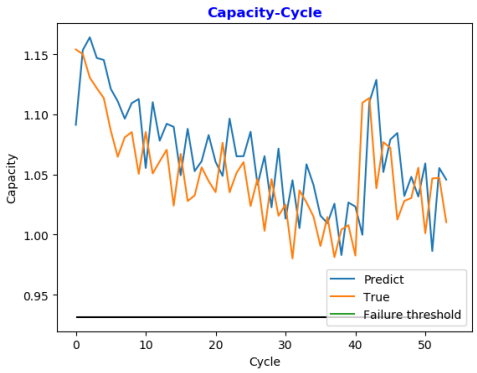
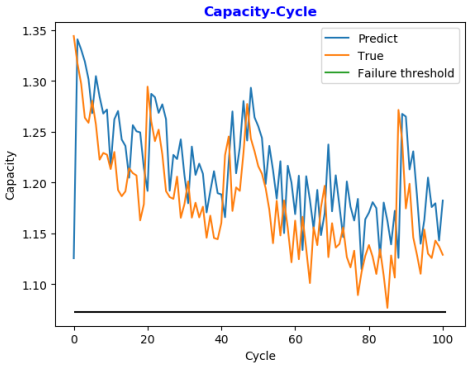


图 26　54 号电池预测误差变化

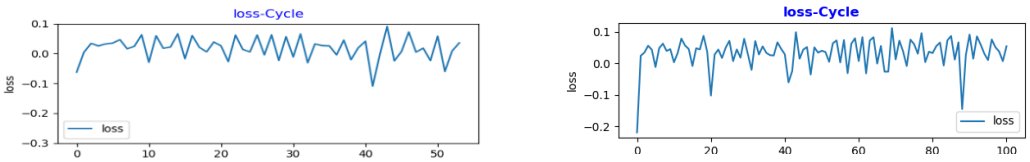
图 25 和图 26 是 54 号电池的预测结果，误差在 [-0.05，0.05] 之间，前期误差较大，后期比较平稳。



（a）#53



（b）#56



#53 误差　　#56 误差

图 27　53 和 56 号电池预测结果与误差变化

图 27 是 53，56 号电池的预测结果，#53 由于数据量比较少，所以都没有到达阈值，#56 起始容量较大，在给定数据 100 个循环内也没有到达阈值，模型在这 2 节电池上特别是 56 号电磁上没有表现出前面 2 个样本较好的收敛效果，原因是 56 号电池震荡较大，只拟合出大概的趋势，误差比较大，可以通过滤波算法或拟合方法改进，本文不再往下研究。

3.5 贝叶斯自动调参

3.5.1 贝叶斯调参步骤

3.5.1.1 理论介绍

常用的调参方法主要有手动调参和自动调参。

手动调整神经网络模型超参数是一项繁琐且费时费力的工作，研究人员在调整模型的超参数时，通常以手动调参为主，调参依据是以往的模型设计经验。然而，手动调参过于依赖研究人员的经验，甚至是运气，对于新手人员来说寻找最优超参数组合难度较大，而且一般模型运行的时间较长，极大地耗费研究人员的精力和时间。因此，人们开发出了自动调参的方法。

直接影响神经网络模型的性能的参数有很多，比如神经网络的隐藏层数、学习率（learning rate）、批量大小（batch）、层数、dropout 和正则化系数等超参数，如何调整这些超参数是一个非常繁琐的工作，又非常重要。目前，手动调优是最常用的方法。即开发人员选择超参数的方式是基于建模经验的，选择“感觉合理”的超参数，然后后续细微的调整是根据模型在数据集上的表现。自动调优，例如随机过程，仍然需要非常大量的计算，并且效率相对较低[39]。然而，近年来超参数的研究取得了很大进展，利用强化学习、遗传算法和神经网络搜索，研究人员也在寻找一种既能提高效率，又能准确调参的方法。

神经网络超参数寻优的方法主要有：网格搜索（穷举），随机搜索（速度快，准确度不高）和贝叶斯优化（基于贝叶斯理论）。

1）网格搜索（Grid Search）

是穷举的思想：将所有可能的参数组合都试一次，是一种简单粗暴的方法，最终会找到最优解，但是非常耗时。 就像在数组中找到最大值一样。 网格搜索的名字的由来是，以具有两个参数的模型举例子。参数 a 具有 5 种选择，参数 B 具有 6 种选择。列出所有可能性并将它们表示为 5* 6 表。每个表格的单元都是一个网格。循环过程是遍历和搜索每个网格，因此称为网格搜索[40]。

在参数组合列表中进行穷举搜索，对所有情况都进行训练，能找到最优的参数；由以上定义可知，这种方法可以找到最优解，但是比较耗时。

2）随机搜索（Random Search）

在寻找最优超参数值时，需要一些数据提前确定。首先，要有一个目标函数，。然后，超参数组合的搜索范围需要确定，搜索范围需要有一个区间，有区间上下限。搜索算法也会有一部分参数，比如搜索的步长，步长的取值过大过小都会影响寻优的结果。

随机搜索的概念类似于网格搜索，会在搜索范围内随机取样。通常，随机搜索比网格搜索更快速，因为随机采样搜索范围。但是，随机搜索的准确度无法保证。

3）贝叶斯搜索（Bayesian Optimization）

贝叶斯优化的名称由来，是因为运用了贝叶斯的思想理论。在每次选择下一个超参数组合时，会考虑之前的参数组合的结果，即在已有结果的基础上选择下一组超参数。

本节重点介绍贝叶斯优化的在 Python 中的一个实现，一个封装好的 Python 模块，名为 hyperopt。

与随机或网格搜索相比，贝叶斯方法跟踪过去的评估结果，他们使用这些结果形成概率模型，将超参数映射到目标函数的得分概率：

$$P\left(score \mid hyperparameters\right)$$

3.5.1.2 贝叶斯优化方法的算法实现

本节的贝叶斯优化方法包括四个部分，分别是目标函数，搜索空间，优化算法和可视化结果与过程。

1）目标函数

目标函数可以是返回我们想要最小化的实际值的任何函数，如函数的返回值 MAE，MSE 等，但是模型只能最小化目标函数的返回值，所以如果我们有一个我们想要最大化的值，例如准确性，那么我们只需让我们的函数返回该指标的负数。本文中目标函数是案例一中的 LSTM 神经网络模型，返回值为训练集的 MAE。

2）搜索空间

搜索空间是我们要搜索的输入值。需要制定分布类型，常用的分布有离散均匀分布，连续均匀分布，正态分布等。本文中的搜索空间是超参数中的学习率，在（0.0005，0.0015）区间内服从连续均匀分布。

3）搜索算法

搜索算法是贝叶斯优化方法的核心算法，目前有两种选择，随机搜索和 Tree of Parzen Estimators (TPE)，本文的方法选择 TPE 算法。

4）可视化过程与结果。

如果能看到调参模型这个“黑匣子”内发生了什么是非常好的。如 Trials 对象能够做到这一点。

3.5.2 现有结果分析

用贝叶斯优化方法自动调整神经网络的学习率，迭代次数为 50，目标是使 MAE 最小化。每次迭代算法给学习率的取值如图 28 所示。

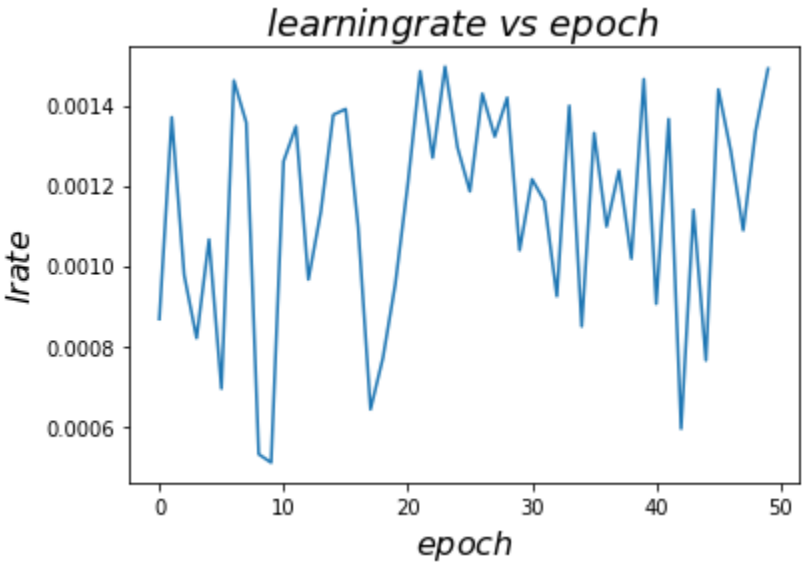


图 28　学习率的取值

由图 28 可知，学习率的取值趋势是上下波动，学习率的最优解是 0.001226，随着迭代次数的逐渐增加，学习率也随之逐渐趋于最优解，并且在第 37 次达到最优解，这也体现了贝叶斯优化的思想，在第 37 迭代之后学习率的取值出现了震荡，算法这样做的目的是防止落入区间内的局部最优解。。

图 29 是学习率和 MAE 的散点图。

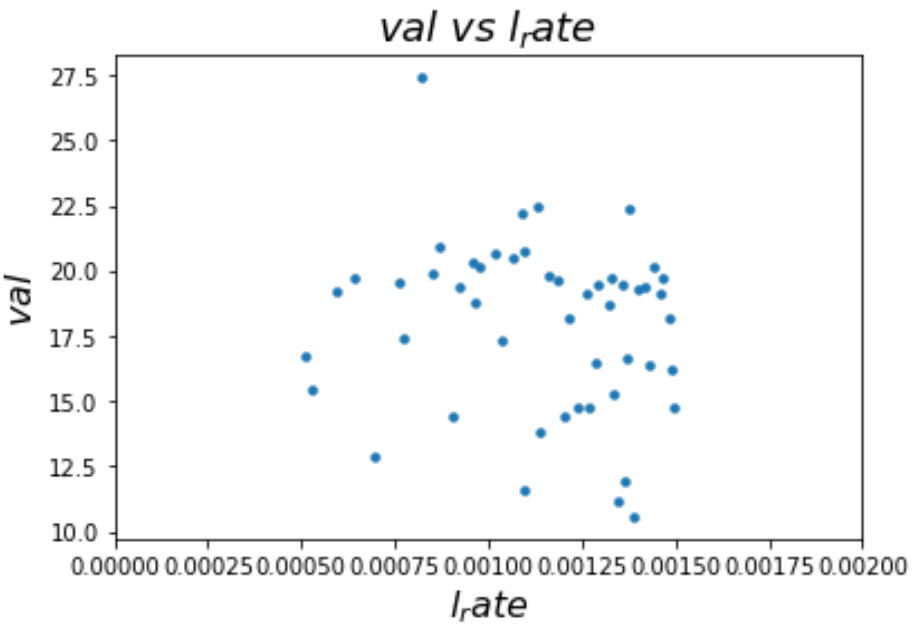


图 29　学习率和 mae 散点图

结果显示在学习率为 0.0012260442375009789 时 MAE 最小，最小值为 10.55162912133929，是一个比调参之前更好的结果。该学习率代入 LSTM 模型。

将该人工网络在训练集上训练 epoch=100 遍，批次大小（batch size）为单台发动机的数据记录的个数，优化器 ADAM，学习率为 0.0012260442375009789。

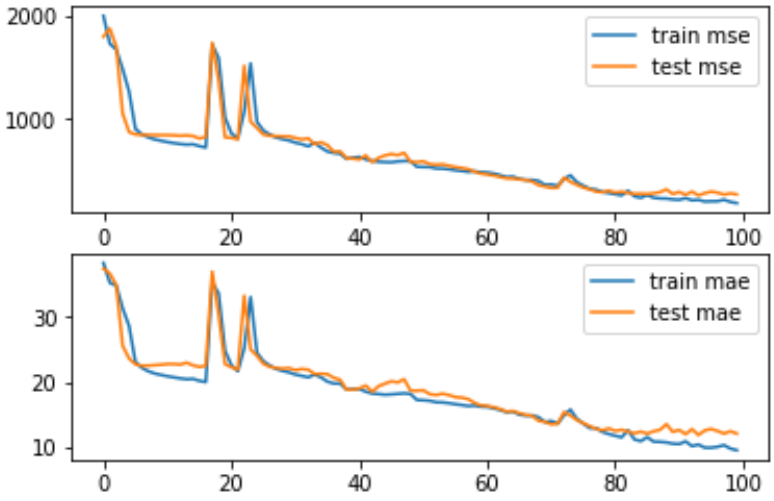
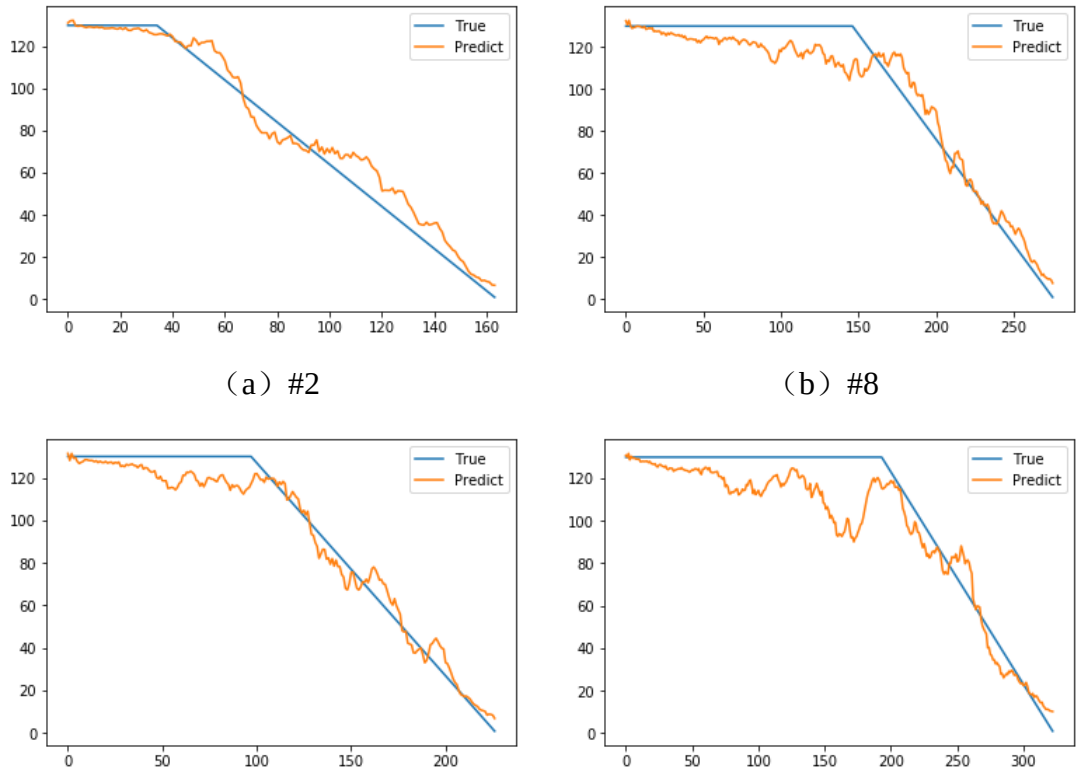


图 30　LSTM epoch MSE 变化图

将上述模型在编号为#2，#8，#111 和#147 样本上进行预测的结果如图 31。



（a）#2　　（b）#8

（c）#111　　（d）#147

图 31　LSTM 预测结果（部分样本）

如图 30 和图 31 所示，优化的 LSTM 已经可以从新数据集中获取相对较好的寿命趋势。在训练集上的平均绝对误差（MAE）为 10.55，由图 36 可知，模型在测试集的四台发动机拟合效果比较好，验证集输出的结果上传到 NASA 的数据预测中心，得到的分数（Score）为 1512.54 分，排进前 15 名，是一个相对较好的得分。可见，贝叶斯优化对于 LSTM 模型的超参数调优是非常有帮助的。

3.6 本章小结

1）利用一种多层 LSTM 模型对多变量时间序列进行预测。

对于高维的海量的时序数据，利用 LSTM 模型得到数据和预测值的映射，搭建预测模型，误差为 13.34，取得良好的预测结果。

2）研究一种基于贝叶斯理论的模型优化方法。

预测模型参数众多，在参考多种参数优化理论基础上，利用基于贝叶斯理论的模型自动优化方法对模型进行参数寻优，并对 LSTM 模型进行了优化，结果使得模型的误差从 13.34 降低到 10.55，优化效果良好。。

3）分别用涡轮发动机的运行数据和锂电池数据做成两个案例，证明 LSTM 模型在不同分析对象上表现同样良好，说明 LSTM 模型的普适性较强。

第四章　基于 Conv-LSTM 的 RUL 预测

对于通用序列建模，在各种先前的研究中，LSTM 作为一种特殊的 RNN 结构已被证明是稳定且强大的，可以对远程依赖性进行建模。LSTM 的主要创新之处在于它的存储单元C_t，它实质上是状态信息的累加器。通过几个自参数化的控制门可以访问，写入和清除该单元。每次有新输入时，如果输入门被激活，则其信息将累积到单元中。此外，如果忘记门 f_t 开启，则过去的单元状态C_t1 可能在此过程中被“忘记”。通过输出门o_t 进一步控制最新单元输出C_t是否将被传播到最终状态H_t。使用存储单元和门控制信息流的一个优势是梯度将被困在单元中（也称为恒定误差圆盘传送带），并防止梯度消失得太快，这对于预测来说是一个关键问题 RNN 模型[61]。 FC-LSTM 可以看作是 LSTM 的多元版本，其中输入，单元输出和状态都是一维向量。在本文中，我们遵循 FC-LSTM 的公式。关键方程式在下面的公式（4.1）中显示，其中∘表示 Hadamard 乘积：

$$\begin{aligned} i_t &= \sigma\left(W_{xi} * \chi_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i\right) \\ f_t &= \sigma\left(W_{xi} * \chi_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f\right) \\ C_t &= f_t \circ C_{t-1} + i_t \circ \tanh\left(W_{xc} * \chi_t + W_{hc} * H_{t-1} + b_c\right) \\ o_t &= \sigma\left(W_{xo} * \chi_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o\right) \\ H_t &= o_t \circ \tanh\left(C_t\right) \end{aligned} \tag{4.1}$$

现在展示的 Conv-LSTM 网络虽然 LSTM 层在处理时间相关方面已经被证明是强大的，但它包含了太多的空间数据冗余为了解决这个问题，提出了一个在输入到状态和状态到状态转换中都具有卷积结构的 LSTM 的扩展。通过叠加多个 Conv-LSTM 层，形成编码预测结构，可以建立更一般的时空序列预测问题的网络模型。

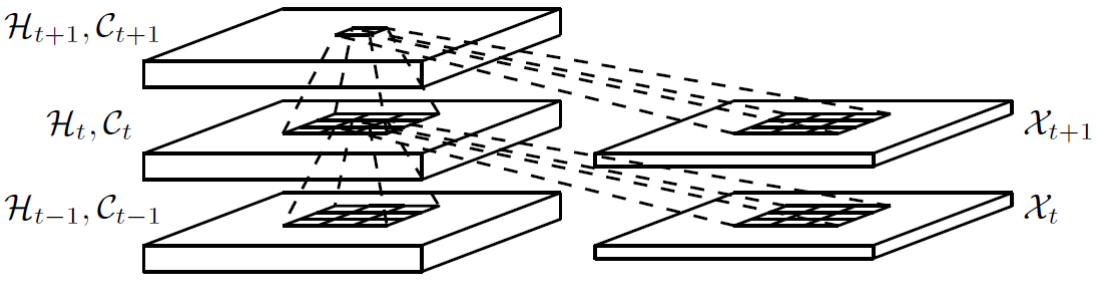


图 32　Conv-LSTM 的结构

4.1 卷积 LSTM

4.1.1 LSTM 常见变体

LSTM 主要的变体如表

表 26　LSTM 主要变体

序号	变体名	特点
1	No Input Gate　(NIG)	没有输入门
2	No Forget Gate　(NFG)	没有遗忘门
3	No Output Gate　(NOG)	没有输出门
4	No Input Activation Function (NIAF)	没有输入激活函数（也就是没有输入门对应的 tanh 层）
5	No Output Activation Function (NOAF)	没有输出激活函数（也就是没有输出门对应的 tanh 层）
6	Peephole connections	直译窥视连接，让门层也会接受细胞状态的输入
7	Coupled Input and Forget Gate (CIFG)	耦合遗忘和输入单元，遗忘门与输入门结合
8	Full Gate Recurrence　(FGR)	FGR 通过增加 LSTM 中所有门之间的递归连接，它增加了 9 个递归权值矩阵，因此大大增加了参数量
9	Bi-LSTM（双向 LSTM）	充分利用上下文信息
10	GRU（门控神经单元）	简化结构，只有更新门和重置门，收敛速度快，适合小样本数据集
11	Conv-LSTM（卷积 LSTM）	加入了卷积的操作，可作为更加复杂模型的基石

从表 26 可知，前 6 种比较好理解，就是去掉了 LSTM 的部分单元。第 7 种是令 input gate（输入门）和 forget gate（遗忘门）加起来和等于 1，类似于 GRU（门控神经单元）的设计思路。第 8 种是类似于 peepholes 的思路，在三个门之间彼此建立连接，这样会增加 3*3=9 个权值矩阵，大大增加了模型参数。

经过调研，结论主要有三点：

1）令 input gate 和 forget gate 加起来和等于 1，或者去掉 peepholes 连接可以简化模型且不会影响结果。这可能也是现在大家用的 LSTM 都没用 peepholes 的原因吧。

2）forget gate 和 output activation function（输出激活函数）对结果影响很大。没有 forget gate，也就是 memory（记忆模块）没有起到作用。Output activation 的非线性映射增加了模型的表达能力。

3）full gate recurrence 不能提高效果。既然 peephole 对结果提高不大，full gate recurrence 当然也不会提高结果。

4）Conv-LSTM 由于在 LSTM 网络层中加入了卷积运算，在处理复杂数据方面更有优势。

综合考虑 LSTM 的各种变体，最终选用 Conv-LSTM 模型做预测以及与 LSTM 模型的对比。

4.1.2 Conv-LSTM 原理

LSTM 的缺点是，在处理数据时，只考虑了时间序列特征，有输入到输出的映射，状态 t 到状态 t+1 的转换，但是没有考虑到数据的空间特征，这些映射和转换也没有涉及到空间特征的转换。 为了克服这个问题，我们设计的一个显着特征是所有输入 $X_1,\ldots, X_t$，单元格输出 $C_1,\ldots, C_t$，隐藏状态 $H_1,\ldots, H_t$，并对其进行相应的门操作；Conv-LSTM 的其中一个是 3D 张量，其最后两个维度是空间维度（行和列）。 为了更好地了解输入和状态，可以将它们想象为站在空间网格上的向量。 Conv-LSTM 的关键方程式在下面的公式中显示，其中*表示卷积算符，而∘像以前一样表示 Hadamard 乘积[62]：

$$\begin{aligned} i_t &= \sigma\left(W_{xi} * \chi_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i\right) \\ f_t &= \sigma\left(W_{xi} * \chi_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f\right) \\ C_t &= f_t \circ C_{t-1} + i_t \circ \tanh\left(W_{xc} * \chi_t + W_{hc} * H_{t-1} + b_c\right) \\ o_t &= \sigma\left(W_{xo} * \chi_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o\right) \\ H_t &= o_t \circ \tanh\left(C_t\right) \end{aligned} \tag{4.2}$$

如果把状态作为移动物体隐藏的表示，具有较大的过渡内核 Conv-LSTM 应该能够捕捉到更快的动作，而一个具有较小的内核可以捕捉慢运动。同样，如果我们采用相似的观点，则由公式（4.2）表示的传统 FC-LSTM 的输入，单元输出和隐藏状态也可以视为 3D 张量，最后两个维度为 1。从这个意义上讲，LSTM 实际上是 Conv-LSTM 的特例，所有功能都位于一个单元中。

为确保状态与输入具有相同的行数和相同的列数，在应用卷积操作之前需要填充。在这里，可以将边界状态上的隐藏状态填充视为使用外部世界的状态进行计算。通常，在第一个输入到来之前，将 LSTM 的所有状态初始化为零，这对应于未来的“完全无知”。类似地，如果我们对隐藏状态执行零填充（本文中使用的填充），则实际上是将外部世界的状态设置为零，并且不假设对外部有先验知识。通过填充状态，可以对边界点进行不同的处理，这在许多情况下很有用。例如，假设正在观察的系统是一个被墙包围的移动

球。尽管看不到这些墙，但可以通过发现球一次又一次地弹跳来推断它们的存在，如果边界点具有与内部点相同的状态转换动力学，则很难做到。

与 LSTM 一样，Conv-LSTM 也可以用作更复杂结构的构建块，对于时空序列预测问题，使用图 33 所示的结构，它由两个网络组成，一个编码网络和一个预测网络，与在文献[62]中一样，预测网络的初始状态和单元输出是从编码网络的最后状态复制而来的两个网络都是通过叠加几个 Conv-LSTM 层形成的由于我们的预测目标与输入具有相同的维数，将预测网络中的所有状态连接起来，并将它们输入到 1×1 卷积层中生成最终预测。

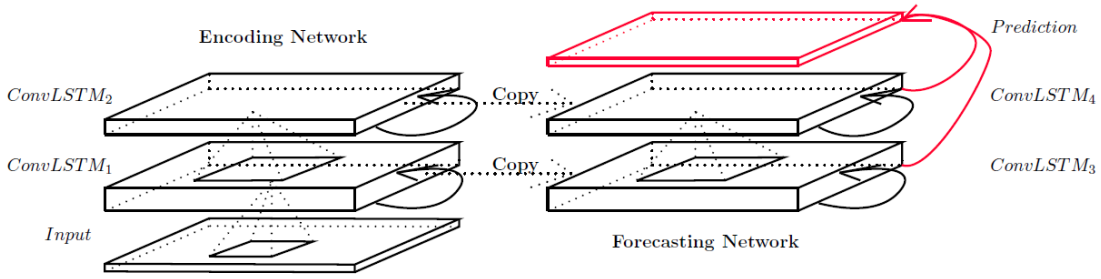


图 33　Conv-LSTM 做 RUL 预测

可以用类似于文献[62]的观点来解释这种结构。编码 LSTM 将整个输入序列压缩为一个隐藏状态张量，预测 LSTM 展开该隐藏状态以给出最终预测：

$$\begin{aligned} \chi_{t+1}, \cdots, \chi_{t+K} &= \underset{\chi_{t+1}, \cdots, \chi_{t+K}}{\arg\max}\ p\left(\chi_{t+1}, \cdots, \chi_{t+K} \mid \chi_{t-J+1}, \chi_{t-J+2}, \cdots, \chi_{t}\right) \\ &\approx \underset{\chi_{t+1}, \cdots, \chi_{t+K}}{\arg\max}\ p\left(\chi_{t+1}, \cdots, \chi_{t+K} \mid f_{encoding}\left(\chi_{t-J+1}, \chi_{t-J+2}, \cdots, \chi_{t}\right)\right) \\ &\approx g_{forecasting}\left(f_{encoding}\left(\chi_{t-J+1}, \chi_{t-J+2}, \cdots, \chi_{t}\right)\right) \end{aligned} \quad (4.3)$$

这种结构也类似于文献[61]中的 LSTM 未来预测模型，只是输入和输出元素都是保留所有空间信息的 3d 张量。由于该网络具有多个层叠的转换层，因此具有很强的代表性，适合于研究的 RUL 预测问题等复杂动力系统的预测。

4.2　RUL 预测

4.2.1 数据

本节所用原始数据为 NASA 数据中心的涡轮发动机仿真数据，利用第二章中介绍的预处理方法，对数据作去中心化和归一化处理。因为 Conv-LSTM 模型在训练过程中可以提取数据的空间特征，因此无需作特征提取。以 3 号发动机为例，处理过的数据如表所示。

4.2.2 模型搭建

建立多层 Conv-LSTM 模型，训练轮次为 40，模型结构如图 34。

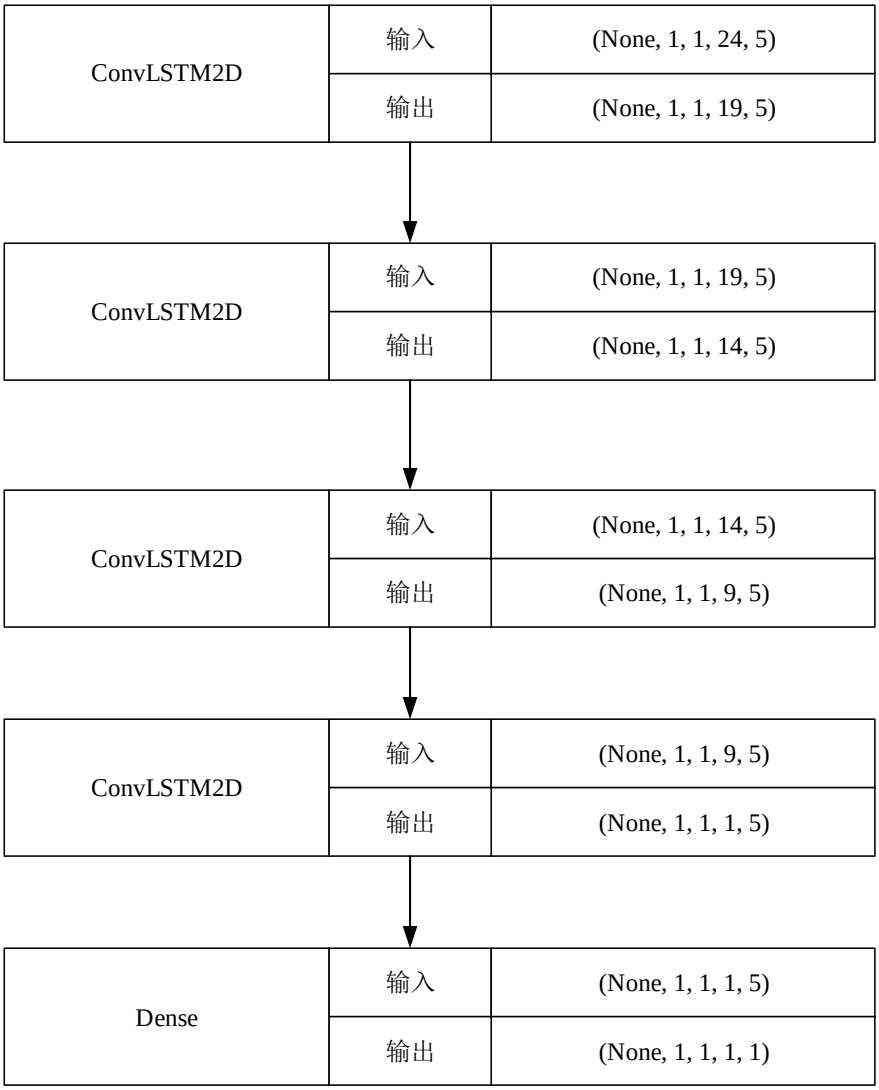


图 34　Conv-LSTM 模型结构

模型共有 5 层，4 层 Conv-LSTM 层和 1 层全连接层，每一层卷积核（filter）个数为 5 个，卷积核的大小为：前 3 层是（1，6），第 4 层为（1，9），学习率为 0.001，训练轮次为 40。模型总的参数个数为 5086 个。

所用数据为涡轮发动机的数据，因为 Conv-LSTM 模型能自己提取数据的空间特征，因此不需要对数据进行特征提取，将数据去中心化和归一化即可。节省了特征提取的工作时间。

训练过程中的误差变化如图 35 所示。

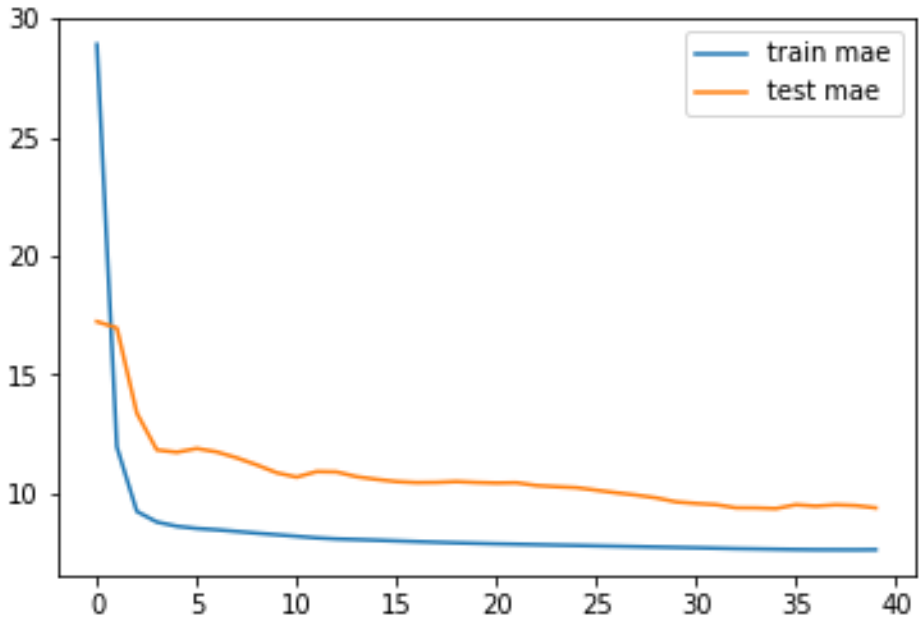
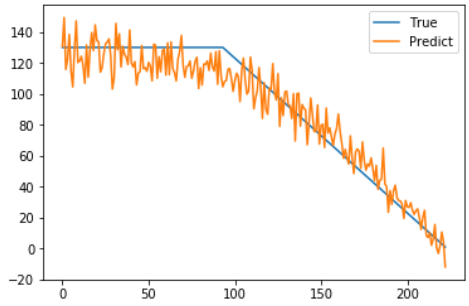
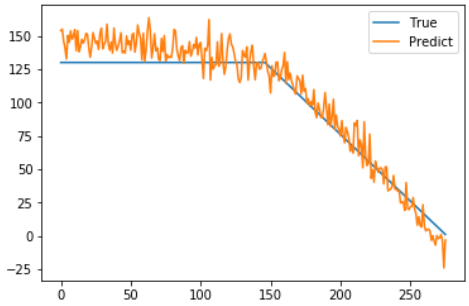


图 35　训练和预测过程中误差变化

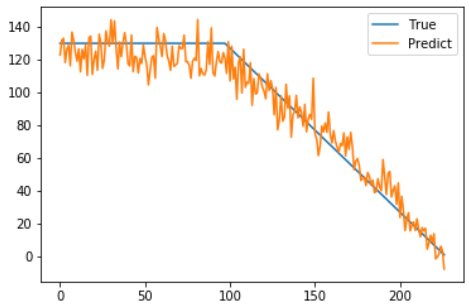
模型在训练集的误差为 5.03，在测试集的训练误差为 11.03，相比 LSTM 误差相对较小。



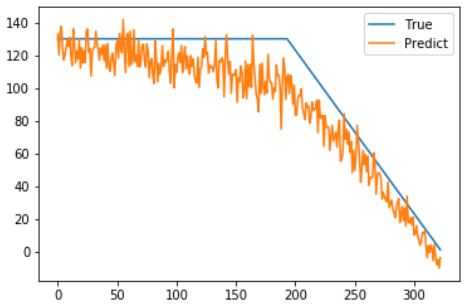
（a）#2



（b）#8



（c）#111



（d）#147

图 36　模型在 4 个样本的预测结果

图 36 中是模型在预测集中选取的 4 个样本的预测情况，分别是 2 号，8 号，111 号和 147 号发动机。由图可知，模型已经基本学习到寿命变化的趋势，结果比较准确。

4.2.3 两种 LSTM 模型的比较

表 27　两种 LSTM 模型对比

指标	LSTM	Conv-LSTM
训练时间/分钟	50	30
训练轮次	100	40
测试集误差 MAE	16.30	11.03
超参数个数	32137	5086
特征提取	PCA 特征提取	无，模型可提取空间特征

由表 27 可知，Conv-LSTM 在误差，训练轮次，训练时间等方面优于 LSTM，对于 LSTM 在时间序列预测方面的应用提供了参考。

4.3 本章小结

利用一种多层 Conv-LSTM 预测了发动机的剩余使用寿命，所用数据为 NASA 的发动机数据，做了去中心化和归一化处理，没有做特征提取。相比普通的 LSTM 模型，Conv-LSTM 模型的训练轮次，训练时间，预测误差相对较小，而且模型本身可以提取数据的空间特征，因此省去了前期数据预处理阶段的特征提取的工作，在时间序列预测方面，在 LSTM 的众多变体中是一个非常好的选择。

总结与展望

总结

通过开题以来的研究，论文已经研究出基于长短期记忆网络（LSTM）的时间序列预测模型，并以此为基础，利用 PCA 算法对数据进行特征提取，将降维后的输入模型，对发动机的剩余使用寿命进行预测，以及利用贝叶斯优化方法自动调整模型的超参数，并将 LSTM 与一种变体 Conv-LSTM 预测方法比较。

1）利用一种 PCA 方法对数据进行特征提取。

针对高维的监测数据，为了方便后续的模型训练预测，提高效率和准确度，利用 PCA 算法对数据进行降维处理，根据特征值的大小选择降维后的维度和特征，提取出数据的主要特征；由于数据不同参量的量级差别很大，通常模型对数据比较敏感，量级很小的参量容易被模型忽略，如果这个参量很重要，就会出现问题，因此对数据做中心化和归一化处理。相比直接将原始数据输入模型的结果，数据预处理后的结果精度明显提升，证明了数据预处理对预测结果很有帮助。

2）利用一种多层 LSTM 模型预测剩余使用寿命。

对于高维的海量的传感器数据，利用 LSTM 及其变体分别得到传感器数据和剩余使用寿命 RUL 的映射，搭建 4 层的预测模型，包括 3 层 LSTM 层和 1 层全连接层，在案例一种用 NASA 数据中心的涡轮发动机数据进行预测，在案例二中，利用某公司的锂电池的实验数据进行预测，适当调整超参数，模型在不同数据上表现同样良好，验证了模型的普适性比较好，通过一套成熟的模型设计方案，可以对不同的时间序列数据进行预测分析。

3）研究一种基于贝叶斯理论的模型优化方法。

预测模型参数众多，手动调参是一项非常繁琐的工作，在自动调参方法里，网络搜索比较耗费时间，随机搜索往往准确度不足，在参考多种参数优化理论基础上，利用基于贝叶斯理论的模型自动优化方法对模型进行参数寻优，贝叶斯方法与其他调参方法最大的不同是会充分利用“历史信息”，即会考虑采用过的参数组合，在参考过去结果的基础上决定下一组超参数的选择。并在模型上进行验证，降低了模型的误差，证明了贝叶斯调参方法的有效性。

4）利用一种多层 Conv-LSTM 模型预测剩余使用寿命。

LSTM 模型在处理时间序列问题时，通常会考虑输入输出之间的映射关系及 t 时刻

和 t+1 时刻的状态转变，不会考虑各个参量之间的联系和影响。在 LSTM 的众多变体中，Conv-LSTM 模型在 LSTM 中加入了卷积操作，用于提取数据的空间特征，因此，用 Conv-LSTM 做预测时，在数据预处理时不需要特征提取，节省了部分工作时间。用 NASA 数据中心的数据进行预测，证明 Conv-LSTM 模型相比 LSTM 模型提升了准确度并且降低了训练时间。

展望

本文在研究基于数据驱动 PHM 技术，结合深度学习算法，提出基于 LSTM 和 Conv-LSTM 的剩余使用寿命预测实施方案和基于 SOM 的健康评估实施方案。由于时间原因和个人水平有限，许多工作细节需要进一步探究。

1）在利用 Conv-LSTM 预测系统的 RUL 时，预测值出现了上下震荡的情况，出现震荡的原因还不明确。有文献提出了用指数平滑的方法改进结果，由于时间关系，本文没有继续研究。

2）在系统的寿命预测方案中，不同系统之间存在差异，而且系统的运行过程中，随着系统的退化，预测和评估结果精度有可能会下降，如何在预测和评估系统中加入自动更新模型的算法，如何让模型动态调整，在线优化超参数，不断更新迭代，保持预测和评估的准确度，将是非常需要关注的研究方向。

参考文献

[1] 张宝珍，曾天翔. PHM:实现 F-35 经济可承受性目标的关键使能技术[J]. 航空维修与工程, 2005(06):20-23

[2] 吕琛，马剑，王自力. PHM 技术国内外发展情况综述[J]. 计算机测量与控制, 2016,24(09):1-4

[3] 景博，杨洲，张劼，等. 故障预测与健康管理系统验证与确认方法综述[J]. 计算机工程与应用, 2011,47(21):23-27

[4] 张宝珍. 国外综合诊断、预测与健康管理技术的发展及应用[J]. 计算机测量与控制, 2008(05):591-594

[5] 崔韦达，李泽滔. 基于定量知识数据驱动故障诊断方法的研究综述[J]. 新型工业化, 2018,8(09):69-72

[6] 陈华伟，刘国平，涂海宁，等. 数据驱动的制造系统快速建模技术[J]. 河北科技大学学报, 2014,35(06):504-511

[7] 彭宇. 数据驱动故障预测和健康管理综述[J]. 机械工程学报, 2009

[8] 赵申坤，姜潮，龙湘云. 一种基于数据驱动和贝叶斯理论的机械系统剩余寿命预测方法[J]. 机械工程学报, 2018,54(12):115-124

[9] 李景奎，段飞飞，蔺瑞管，等. 飞机整体驱动发电机可靠性评估及寿命预测[J]. 中国工程机械学报, 2018,16(05):399-403

[10] 何世禹. 航天器在轨寿命预测与可靠性评价[J]. 航天器环境工程, 2008(03):209-211

[11] 厉海涛，金光，周经伦，等. 动量轮维纳过程退化建模与寿命预测[J]. 航空动力学报, 2011,26(03):622-627

[12] 陶耀东，李宁. 基于 ARIMA 模型的工业锂电池剩余使用寿命预测[J]. 计算机系统应用, 2017,26(11):282-287

[13] 王有元. 基于关联规则分析的电力变压器故障马尔科夫预测模型[J]. 航空动力学报, 2010

[14] Valpola H. From neural PCA to deep unsupervised learning[J]. Eprint Arxiv, 2014

[15] Jung S, Marron J S. PCA consistency in high dimension, low sample size context[J]. Annals of Statistics, 2007,37(6B):4104-4130

[16] Feng J, Xu H, Yan S. Online robust PCA via stochastic optimization[J]. Advances in Neural Information Processing Systems, 2013,26

[17] Soule A, Soule A, Diot C, et al. Sensitivity of PCA for traffic anomaly detection: IEEE

Transactions on Instrumentation & Measurement, 2007[C]

[18] Malhi A, Gao R X. PCA-based feature selection scheme for machine defect classification[J]. IEEE Transactions on Instrumentation & Measurement, 2010,53(6):1517-1525

[19] Destefanis G, Barge M T, Brugiapaglia A, et al. The use of principal component analysis (PCA) to characterize beef[J]. Meat Science, 2000,56(3):255-259

[20] Strapp J W, Leaitch W R, Liu P S K. Hydrated and Dried Aerosol-Size-Distribution Measurements from the Particle Measuring Systems FSSP-300 Probe and the Deiced PCASP-100X Probe[J]. Journal of Atmospheric & Oceanic Technology, 1992,9(5):548-555

[21] Chennubhotla C, Jepson A. Sparse PCA Extracting Multi-scale Structure from Data[J]. Eprint Arxiv, 2001

[22] Desale R P, Verma S V. Study and analysis of PCA, DCT & DWT based image fusion techniques: Signal Processing Image Processing & Pattern Recognition (ICSIPR), 2013 International Conference on, 2013[C]

[23] Fulda G J, Giberson F, Fagraeus L. A PROSPECTIVE RANDOMIZED TRIAL OF NEBULIZED MORPHINE COMPARED TO PCA MORPHINE IN THE MANAGEMENT OF ACUTE THORACIC PAIN[J]. The Journal of Trauma: Injury, Infection, and Critical Care, 2009,57(2):444

[24] Lanorte A, Manzi T, Nolè G, et al. On the Use of the Principal Component Analysis (PCA) for Evaluating Vegetation Anomalies from LANDSAT-TM NDVI Temporal Series in the Basilicata Region (Italy)[M]. 2015

[25] Lanorte A, Manzi T, Nolè G, et al. On the Use of the Principal Component Analysis (PCA) for Evaluating Vegetation Anomalies from LANDSAT-TM NDVI Temporal Series in the Basilicata Region (Italy)[M]. 2015

[26] Nguyen T D, Gupta S, Rana S, et al. Cascade Bayesian Optimization[J]. 2016

[27] Baptista R, Poloczek M. Bayesian optimization of combinatorial structures: International Conference of Machine Learning, 2018[C]

[28] Chang W A, Ramakrishna R S, Goldberg D E. Real-coded Bayesian Optimization Algorithm[M]. Springer Berlin Heidelberg, 1970

[29] Soltan Ghoraie L. Bayesian Optimization Algorithm for Non-unique Oligonucleotide Probe Selection, 1997[C]

[30] Pautrat R, Chatzilygeroudis K, Mouret J B. Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search: 2018 IEEE International Conference on

Robotics and Automation (ICRA), 2018[C]

[31] McIntire M, Cope T, Ratner D, et al. Bayesian Optimization of FEL Performance at LCLS, 2016[C]

[32] Nguyen V, Gupta S, Rane S, et al. Bayesian Optimization in Weakly Specified Search Space: 2017 IEEE International Conference on Data Mining (ICDM), 2017[C]

[33] Flanagan J A. Self-organisation in Kohonen's SOM[J]. 1996,9(7):1185

[34] 陈佳, 刘冬雪, 武大硕. 基于特征选取与 LSTM 模型的股指预测方法研究[J]. 计算机工程与应用, 2019,55(6):108-112

[35] 李高盛, 彭玲, 李祥, 等. 基于 LSTM 的城市公交车站短时客流量预测研究[J]. 公路交通科技, 2019,36(02):128-135

[36] 王鑫, 吴际, 刘超, 等. 基于 LSTM 循环神经网络的故障时间序列预测[J]. 北京航空航天大学学报, 2018,44(04):772-784

[37] 彭燕, 刘宇红, 张荣芬. 基于 LSTM 的股票价格预测建模与分析[J]. 计算机工程与应用, 2019,55(11):209-212

[38] 李鹤喜, 韩新乐, 方灶军. 一种基于 CNN 深度学习的焊接机器人视觉模型[J]. 焊接学报, 2019,40(02):154-160

[39] 王莉. 基于 SOM 自组织神经网络的企业信用评估模型[D]. 太原理工大学, 2005

[40] 姜晨, 徐廷学, 余仁波. 基于 SOM 网络的军械维修器材供应链绩效评估[J]. 兵工自动化, 2011,30(05):21-25

[41] 徐廷学, 陈红, 周东君, 等. 基于 SOM 网络的军事供应链绩效评估[J]. 海军航空工程学院学报, 2011,26(03):351-355

[42] 赵伟. 高斯的 SOM 神经网络在雷达抗干扰效能评估中的应用[J]. 统计与决策, 2010

[43] 周璞. 自组织映射神经网络 SOM 法对江淮流域逐日降水量的模拟评估[J]. 统计与决策, 2001

[44] 张浩, 赵莉华, 景伟, 等. 基于 Relief 特征量优化及 SOM 网络的断路器操作机构状态评估[J]. 高压电器, 2017,53(09):240-246

[45] 樊国敬, 田秀华. 基于 SOM 神经网络的区域复合生态系统健康评估[J]. 统计与决策, 2017(11):85-89

[46] 王沁, 曾广平, 涂序彦. 基于复杂网络和 SOM 网的 IT 项目组合风险分析[J]. 计算机仿真, 2018,35(08):434-438

[47] 许逸凡, 李杰, 魏义涛. 基于 SOM 网络的机场天气聚类分析[J]. 数学的实践与认

识, 2016,46(17):210-217

[48] 高志. SOM 神经网络模型在商业银行信用风险评估中的应用研究[J]. 电脑知识与技术(学术交流), 2007(14):498-499

[49] 王鑫, 吴际, 刘超. 基于 LSTM 循环神经网络的故障时间序列预测[J]. 北京航空航天大学学报, 2018(4):772-784

[50] Vautard R , Yiou P , Ghil M . Singular-spectrum analysis: A toolkit for short, noisy chaotic signals[J]. Physica D, 1992, 58(1-4):95-126

[51] Golyandina N , Korobeynikov A . Basic Singular Spectrum Analysis and Forecasting with R[J]. Computational Statistics & Data Analysis, 2014, 71:934–954

[52] Bianco V , Manca O , Nardini S . Electricity consumption forecasting in Italy using linear regression models[J]. Energy, 2009, 34(9):1413-1421

[53] Hanson J , Yang Y , Paliwal K , et al. Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks[J]. Bioinformatics, 2016:btw678

[54] Ma, Xiaolei, Tao, Zhimin, Wang, Yinhai. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data[J]. Transportation Research Part C, 54:187-197

[55] Alex Graves. Long Short-Term Memory[M]// Supervised Sequence Labelling with Recurrent Neural Networks. 2012

[56] Lecun Y, Bengio Y, Hinton G. Deep learning.[J]. 2015, 521(7553):436

[57] Samaneh Misaghi, Omid Sojoodi Sheijani. A hybrid model based on support vector regression and modified harmony search algorithm in time series prediction[C]// 2017 5th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS). IEEE, 2017

[58] Márcio das Chagas Moura, Zio E , Lins I D , et al. Failure and reliability prediction by support vector machines regression of time series data[J]. Reliability Engineering [?] System Safety, 2011, 96(11):1527-1534

[59] Rocco S , Claudio M . Singular spectrum analysis and forecasting of failure time series[J]. Reliability Engineering & System Safety, 2013, 114:126-136

[60] Sapankevych N , Sankar R . Time Series Prediction Using Support Vector Machines: A Survey[J]. IEEE Computational Intelligence Magazine, 2009, 4(2):24-38

[61] Vichare N M , Pecht M G . Prognostics and health management of electronics[M]. IEEE, 2006

[62] Tan, Chao, Feng, Xin, Long, Jianwu. FORECAST-CLSTM: A New Convolutional

LSTM Network for Cloudage Nowcasting[J]

附录

附录包含 4 部分，包括 LSTM 模型代码，贝叶斯代码，PCA 代码，Conv-LSTM 模型代码。

附录 1 LSTM 模型代码

```
# -*- coding: utf-8 -*-
"""
Created on Tue Sep  4 11:19:55 2018

@author: ourda
"""
#%%
#     import every package
##################################

import pandas as pd
import numpy as np
import tensorflow
print('tensorflow: %s' % tensorflow.__version__)
import keras
print('keras: %s' % keras.__version__)
import datetime
from matplotlib import pyplot
from keras import regularizers
from keras.models import Sequential
from keras.layers import Activation, Dense, LSTM, CuDNNLSTM, TimeDistributed,
Dropout, GRU, CuDNNGRU,SimpleRNN
from keras.utils import plot_model
import os
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
from math import sqrt
#import smooth
import striding

np.random.seed(7) #随机数种子，确保结果可以重现，实际上这样还是有些不严谨，或许可以不用种子反复运行十次求平均
#%%

#     initialize global variables
#     models    ####################
epoch_nb = 50 #epoch 数，最多在全部数据上训练 50 遍，可能由于误差不再下降而提前结束训练
maxnb = 130 #这是寿命标签的上限，寿命从 130 开始倒数到 1
timestepstoputin = None #这个是论文里一次输入的时间步数，是下面很多函数的输入条
```

件，实际上不需要它（单步就好），这是早期试验的时候留下的

```
#     folder    ####################
folder = '1205-test' #保存结果的文件夹名称，12 月 5 号，每次修改程序都改这里以确保每次试验都存了结果
try:
    os.makedirs('D:/archive/'+folder+'/logs/')

except:
    print('文件夹已存在')
#     smooth    ####################
alpha = 0.7 #指数平滑的参数
order = 3 #指数平滑的阶数，实际上不用平滑，这两行没啥用


#%%
#     define all function

#keras 里默认没有 rmse，这里定义 rmse 的函数，就是把 mse 开方
def root_mean_squared_error(y_true, y_pred):
        return keras.backend.sqrt(keras.backend.mean(keras.backend.square(y_pred - y_true), axis=-1))

#keras 默认 relu 不能调里面的参数，这里定义以后可以改参数
def relu(x):
    return keras.activations.relu(x, alpha=0.3, max_value=None)

#这个是获取数据的函数，参数 data 为全部数据（预处理之后的），index 就发动机编号，ts 就是 timesteps
def getdata(data, index, ts = None):
    datai = data.loc[index].values #从所有数据中取出编号为 index 的发动机
    data_y = np.arange(datai.shape[0], 0, -1) #刚取出的数据总长，倒数到 1
    data_y[:-maxnb] = maxnb #总长可能超过 130，这里把超过的都等于 130
    data_y = scaler.transform(data_y.reshape(-1, 1)) #归一化
    if ts == None: #到这就完事了，后面的是 ts 的其他情况
        return datai.reshape(1,datai.shape[0],datai.shape[1]), data_y.reshape(1,data_y.shape[0],1)
    elif ts == 1:
        return datai.reshape(datai.shape[0],1,datai.shape[1]), data_y
    else:
        datax_nor = striding.striding(datai, True, ts)
        datax_nor = datax_nor[::-ts]
        datax_nor = datax_nor[::-1]
        data_y = striding.striding(data_y.reshape(data_y.shape[0],1), True, ts)
        data_y = data_y[::-ts]
        data_y = data_y[::-1]
        return datax_nor, data_y
```

```
#这是建立模型的函数，之所以用函数来建立模型是为了能够进行网格搜索，网格搜索也可以用 sklearn 里面的语法做，具体问师弟，这里只能调三层网络的节点数
def buildmodel(ly1u, ly2u, ly3u):
    model = Sequential()
    #model.add(TimeDistributed(Dense(24, activation=ReLU), input_shape=(timestepstoputin, 24)))
    model.add(LSTM(ly1u, input_shape=(timestepstoputin, 24), return_sequences = True, stateful = False
                                ))
#     model.add(Activation('elu'))
#     model.add(Dropout(0.4, noise_shape=None, seed=None))
    model.add(LSTM(ly2u, return_sequences=True))
#     model.add(Activation(relu))
#     model.add(Dropout(0.4, noise_shape=(None, 1, 12), seed=None))
    model.add(LSTM(ly3u, return_sequences=True))
#     model.add(Activation('elu'))
#     model.add(Dropout(0.4, noise_shape=(None, 1, 12), seed=None))
    #model.add(CuDNNLSTM(1, return_sequences=True, stateful=False))
#     model.add(Activation('tanh'))
    model.add(TimeDistributed(Dense(1)))
#     model.add(Activation('elu'))
#     model.add(Activation(relu))
    #model.add(Activation('tanh'))
    model.compile(loss='mse', optimizer=use, metrics=['mae'])
    model.summary()
    return model

#0-1 归一化，用函数是为了配合 187 行的 apply，这里面不太好说，我也是试出来的，用就行了
def scaleIt(data):
    feature_scaler= MinMaxScaler()
    data = np.array(data)
    scaled = feature_scaler.fit_transform(data.reshape(-1, 1))
    return scaled.reshape(scaled.shape[0])

#predict 函数，通过这个获得预计结果，并且输出文件
def predicting(data_test, model, name = 'M1'):
    predict_table = []
    indexArray_test = np.arange(1, data_test.index[-1][0]+1, 1)
    for index in indexArray_test:
        datai_nor, data_y = getdata(data_test, index, timestepstoputin)
        evaluate = model.predict(datai_nor, batch_size = datai_nor.shape[0], verbose=0)
        evaluate_true = scaler.inverse_transform(evaluate.flatten().reshape(-1,1))
#         s1, s2 = smooth.smoothing(alpha, evaluate_true, order)
        predict_table.append(evaluate_true[-1][0])
    np.savetxt('D:/archive/'+ folder +'/'+name+'-result.txt', predict_table, fmt='%.4f')
    return predict_table

#结果举例演示作图
```

```
    '''
def demoshow(model,eg=[2,8,9,10]):
    for index_p in eg:
        datap_nor, datap_y = getdata(data_nor, index_p)
        predict = model.predict(datap_nor, batch_size=datap_nor.shape[0], verbose=0)
        datap_true = scaler.inverse_transform(predict.flatten().reshape(-1,1))
        data_true = scaler.inverse_transform(datap_y.flatten().reshape(-1,1))
#         s1, s2 = smooth.smoothing(alpha, datap_true, order)
        pyplot.plot(data_true, label='True')
        pyplot.plot(datap_true, label='Predict')
#         pyplot.plot(s1, label='s1')
#         pyplot.plot(s2, label='s2')
        pyplot.legend()
        pyplot.show()
        rmse = sqrt(mean_squared_error(datap_true, data_true))
#         rmse1 = sqrt(mean_squared_error(s1, data_true))
#         rmse2 = sqrt(mean_squared_error(s2, data_true))
#         print('rmse = '+ str("%.6f"%rmse) +'    '+str("%.6f"%rmse1)+'
'+str("%.6f"%rmse2))
        '''

#%%
#     setting modules
###################################
#     keras
###################################这一堆是不同的优化器，看情况改 buildmodel
里的设置
use = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0,
amsgrad=False)
RMS = keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)
AdaGrad = keras.optimizers.Adagrad(lr=0.001, epsilon=1e-06)


###################################
#     scaler
###################################0-1 归一化的详细设置，主要是对 y 的归一化，
对 x 的归一化是上面 scaleIt 函数
ScaleToFit = np.arange(130, 0, -1).reshape(-1, 1)
feature_range = (0, 1)
fr = 1/(feature_range[1] - feature_range[0])
scaler = MinMaxScaler(feature_range)
scaler.fit(ScaleToFit)
scalechanger = (ScaleToFit[0]-ScaleToFit[-1])*(ScaleToFit[0]-ScaleToFit[-1])*fr*fr #这是为
了把 keras 默认输出的 0-1 上的误差还原为 1-130 上的误差，所用的参数

###################################
#     K-fold
###################################把数据随机分成 10 份，分训练集测试集用的
kf = KFold(n_splits=10, shuffle=True)
```

```
#%%
#     import data  输入数据
##################################

data_train = pd.read_table('D:/DL/train.txt', delim_whitespace=True, header=None,
                           index_col=[0,1], prefix='X')

data_test = pd.read_table('D:/DL/train.txt', delim_whitespace=True, header=None,
                           index_col=[0,1], prefix='X')

#     std data 0-1 标准化数据，针对 x。y 这里没有，getdata 函数会生成 y
##################################
dataDict = {'train': data_train, 'test': data_test}
alldata = pd.concat(dataDict, axis=0, names =['Series name', 'part ID','time'])
alldata = alldata.apply(scaleIt, axis=0, raw=True)
data_nor = alldata.loc['train']
data_test = alldata.loc['test']

#tenfeatures = ['X6','X7','X8','X11','X15','X16','X19']
#data_nor = pd.DataFrame(data_nor,columns = tenfeatures)
#data_test = pd.DataFrame(data_test,columns = tenfeatures)
#%%
def training(data, model, name):

    setcountdown = 20 #训练误差倒计时起点
    setvcountdown = 30 #测试误差倒计时起点，两个起点固定不会改变
    countdown = setcountdown #初始化倒计时
    vcountdown = setvcountdown #初始化倒计时
    keras.backend.set_value(use.lr, 0.1) #初始化学习率
    loss_result = [] #建立保存用来结果的变量
    mae_result = [] #建立保存用来结果的变量
    v_loss_result = [] #建立保存用来结果的变量
    v_mae_result = [] #建立保存用来结果的变量
    bestloss = 10000. #初值，比较大的数，无意义，用来避免第一轮训练无人可比而报
错
    bestvloss = 10000.

    indexArray = np.arange(1, data.index[-1][0]+1, 1) #发动机编号数组
    splitedIndex = kf.split(indexArray) #把数组按 k 折随机分两份
    trainIndex, testIndex = next(splitedIndex) #获得刚才分好的两份编号

    print(name + '    Training start at ' +
str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
    #创建保存训练过程的文件
    logfile = open('D:/archive/'+folder+'/logs/'+name+'.txt','w')
```

```
    logfile.writelines(name + '\t' + 'Time' + '\t' + 'Epoch' + '\t' + 'Cdown' + '\t' + 'MSE' + '\t' +
'MAE' + '\t' +
                                   'vMSE' + '\t' + 'vMAE' + '\t' + 'LR\r\n')

    #这里正式开始训练
    for i in range(epoch_nb):
        if i % 5 == 0: #这里每五个 epoch 存一次模型
            model.save('D:/archive/'+folder+'/'+ str(datetime.date.today()) + name + ' ' +
str(i) + '_model.h5')

        loss_table = [] #建立保存用来结果的变量，这存的是每一个发动机的结果，上
面的 loss_result 存的是这个求平均
        mae_table = []
        v_loss_table = []
        v_mae_table = []
#         np.random.shuffle(trainIndex) #这里可以打乱每次训练的发动机顺序
        #Train
        for index in indexArray[trainIndex]: #对某个发动机的数据进行训练
            datai_nor, data_y = getdata(data,index,timestepstoputin)
            history = model.fit(datai_nor,
                                data_y,
                                epochs=1,
                                batch_size = datai_nor.shape[0],
                                verbose=0,
                                shuffle=False)

            loss_table.append(history.history['loss'][0]) #保存这一台结果到 loss_table，
循环直到每一台的结果都在 loss_table 里
            mae_table.append(history.history['mean_absolute_error'][0])

        loss_result.append((sum(loss_table)/len(loss_table))*scalechanger) #把 loss_table
里的求平均，作为当前 epoch 在所有训练集数据上的表现
        mae_result.append((ScaleToFit[0]-ScaleToFit[-
1])*(sum(mae_table)/len(mae_table))*fr)

        for index in indexArray[testIndex]: #这是测试集，和前面类似
            datai_nor, data_y = getdata(data,index,timestepstoputin)
            evaluate = model.evaluate(datai_nor,
                                      data_y,
                                      batch_size = datai_nor.shape[0],
                                      verbose=0,
                                      sample_weight=None)
            v_loss_table.append(evaluate[0])
            v_mae_table.append(evaluate[1])

        v_loss_result.append((sum(v_loss_table)/len(v_loss_table))*scalechanger)
        v_mae_result.append((ScaleToFit[0]-ScaleToFit[-
1])*(sum(v_mae_table)/len(v_mae_table))*fr)
#这里到 316 行大概是判断误差是否继续下降，判断倒计时是否满足条件，提前结束训
```

练或是调整学习率，具体参考论文里的判断流程图，另外还有一些训练过程的文字输出，不关键

```
            if loss_result[-1][0] >1000.:
                bestloss = loss_result[-1][0]
                if v_loss_result[-1][0] >1000.:
                    bestvloss = v_loss_result[-1][0]

            if bestloss - 5 > loss_result[-1][0] or bestvloss - 1 > v_loss_result[-1][0]:
                if bestloss - 5 > loss_result[-1][0]:
                    bestloss = loss_result[-1][0]
                    countdown = setcountdown

                if bestvloss - 1 > v_loss_result[-1][0]:
                    bestvloss = v_loss_result[-1][0]
                    countdown = setcountdown
                    vcountdown = setvcountdown
                print(str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')) + '
loss :' + str("%11.6f"% (loss_result[-1])) + '       mae :' + str("%6.2f"%(mae_result[-1])) + '
<' + str("%8.6f"%keras.backend.get_value(use.lr)))
                try:
                    print(' Epoch ' + str("%5d"%(i+1)) + str("%7d"%countdown) + ' - vloss :'
+ str("%11.6f"% (v_loss_result[-1])) + ' - vmae :' + str("%6.2f"%(v_mae_result[-1])) +
str("%12.6f"% (loss_result[-2] - loss_result[-1])))
                except:
                    print(' Epoch ' + str("%5d"%(i+1)) + str("%7d"%countdown) + ' - vloss :'
+ str("%11.6f"% (v_loss_result[-1])) + ' - vmae :' + str("%6.2f"%(v_mae_result[-1])))
                else:
                    vcountdown -=1
                    if vcountdown == 0:
                        break

            else:
                vcountdown -= 1
                if vcountdown == 0:
                        break
                countdown -= 1
                if countdown == 0:
#                     setcountdown += 5
                    keras.backend.set_value(use.lr, 0.5 * keras.backend.get_value(use.lr))
                    if keras.backend.get_value(use.lr) < 0.000250:
                        keras.backend.set_value(use.lr, 0.000250)
                        setcountdown = 10

                print(str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')) + '
loss :' + str("%11.6f"% (loss_result[-1])) + '       mae :' + str("%6.2f"%(mae_result[-1])) + '
<'    + str("%8.6f"%keras.backend.get_value(use.lr)))
                try:
                    print(' Epoch ' + str("%5d"%(i+1)) + str("%7d"%countdown) + '
vloss :' + str("%11.6f"% (v_loss_result[-1])) + '      vmae :' + str("%6.2f"%(v_mae_result[-1]))
+ str("%12.6f"% (loss_result[-2] - loss_result[-1])))
```

```
                except:
                    print(' Epoch ' + str("%5d"%(i+1)) + str("%7d"%countdown) + '
vloss :' + str("%11.6f"% (v_loss_result[-1])) + '      vmae :' + str("%6.2f"%(v_mae_result[-
1])))

            logfile.writelines(name + '\t' +
str(datetime.datetime.now().strftime('%m-%d %H:%M:%S')) +
                               '\t' + str(i+1) + '\t' + str(countdown) +
                               '\t' + str(loss_result[-1]) + '\t' + str(mae_result[-1]) +
                               '\t' + str(v_loss_result[-1]) + '\t' + str(v_mae_result[-1]) +
                               '\t' + str(keras.backend.get_value(use.lr)) +
                               '\r\n')
#             final train msg
#             print(name + str(datetime.datetime.now().strftime('    %Y-%m-%d %H:%M:%S'))
+ '      Epoch ' + str("%5d"%(i+1)) + str("%7d"%countdown) + '       loss :' + str("%8.2f"%
(loss_result[-1])) + '       mae :' + str("%6.2f"%(mae_result[-1])) + '      <'    +
str("%8.6f"%keras.backend.get_value(use.lr)))

            if countdown == - setcountdown:
                print(name + '    Training stopped at ' +
str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')) +
                      '       loss :' + str("%11.6f"% (loss_result[-1])) +
                      '      vloss :' + str("%11.6f"% (v_loss_result[-1])))
                print('      Epoch ' + str("%5d"%(i+1)) + '      <'    +
str("%8.6f"%keras.backend.get_value(use.lr)))

                break

        print(name + '    Training Finished at ' +
str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')) +
              '       loss :' + str("%11.6f"% (loss_result[-1])) + '      vloss :' + str("%11.6f"%
(v_loss_result[-1])))
        print('      Epoch ' + str("%5d"%(i+1)) + '      <'    +
str("%8.6f"%keras.backend.get_value(use.lr)) +
              '       mae :' + str("%6.2f"%(mae_result[-1])) + '      vmae :' +
str("%6.2f"%(v_mae_result[-1])))
        logfile.writelines(name+'Training Finished at ' + '    ' +
str(datetime.datetime.now().strftime('%m-%d %H:%M:%S')))
        logfile.writelines(name +'    Training Finished at ' + '    ' +
str(datetime.datetime.now().strftime('%m-%d %H:%M:%S')))
        logfile.writelines('      Epoch ' + str("%5d"%(i+1))+'\t' + 'bestloss' +'\t' + str(bestloss)+'\t'
+'bestvloss' +'\t' + str(bestvloss))
        logfile.close()

        #       Show result
        print('Result : ')
        pyplot.subplot(211)
        pyplot.plot(loss_result, label='train mse')
        pyplot.plot(v_loss_result, label='test mse')
        pyplot.legend()
```

```
    pyplot.subplot(212)
    pyplot.plot(mae_result, label='train mae')
    pyplot.plot(v_mae_result, label='test mae')
    pyplot.legend()
    pyplot.show()

    return model, pd.DataFrame({'loss_result':loss_result, 'mae_result':mae_result,
                                'v_loss_result':v_loss_result, 'v_mae_result':v_mae_result})

def finaltrain(data, model, name): #这个函数和上面的类似，只是没有测试集，用所有数据训练

    setcountdown = 30
    countdown = setcountdown
    keras.backend.set_value(use.lr, 0.001)
    loss_result = []
    mae_result = []
    bestloss = 10000

    indexArray = np.arange(1, data.index[-1][0]+1, 1)
    splitedIndex = kf.split(indexArray)
    trainIndex, testIndex = next(splitedIndex)

    print(name + '    Training start at ' +
str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')))

    logfile = open('D:/archive/'+folder+'/logs/'+name+'.txt','w')
    logfile.writelines(name + '\t' + 'Time' + '\t' + 'Epoch' + '\t' + 'Cdown' + '\t' + 'MSE' + '\t' +
'MAE' + '\t' +
                       'LR\r\n')


    for i in range(epoch_nb):
        if i % 5 == 0:
            model.save('D:/archive/'+folder+'/'+ str(datetime.date.today()) + name + ' ' +
str(i) + '_model.h5')
        loss_table = []
        mae_table = []
        np.random.shuffle(indexArray)
        #Train
        for index in indexArray:
            datai_nor, data_y = getdata(data,index,timestepstoputin)
            history = model.fit(datai_nor,
                                data_y,
                                epochs=1,
                                batch_size = datai_nor.shape[0],
                                verbose=0,
                                shuffle=False)

            loss_table.append(history.history['loss'][0])
```

```
                mae_table.append(history.history['mean_absolute_error'][0])

            loss_result.append((sum(loss_table)/len(loss_table))*scalechanger)
            mae_result.append((ScaleToFit[0]-ScaleToFit[-
1])*(sum(mae_table)/len(mae_table))*fr)

            if loss_result[-1][0] >1000:
                bestloss = loss_result[-1][0]

            if bestloss - 1 > loss_result[-1][0]:
                bestloss = loss_result[-1][0]
                countdown = setcountdown
            else:
                countdown -= 1
                if countdown == 0:
#                     setcountdown += 5
                    keras.backend.set_value(use.lr, 0.5 * keras.backend.get_value(use.lr))
                    if keras.backend.get_value(use.lr) < 0.000250:
                        keras.backend.set_value(use.lr, 0.000250)
                        setcountdown = 10

            logfile.writelines(name + '\t' +
str(datetime.datetime.now().strftime('%m-%d %H:%M:%S')) +
                                    '\t' + str(i+1) + '\t' + str(countdown) +
                                    '\t' + str(loss_result[-1]) + '\t' + str(mae_result[-1]) +
                                    '\t' + str(keras.backend.get_value(use.lr)) +
                                    '\r\n')
#             final train msg
            print(name + str(datetime.datetime.now().strftime('    %Y-%m-%d %H:%M:%S')) +
'     Epoch ' + str("%5d"%(i+1)) + str("%7d"%countdown) + '       loss :' + str("%8.2f"%
(loss_result[-1])) + '       mae :' + str("%6.2f"%(mae_result[-1])) + '     <'    +
str("%8.6f"%keras.backend.get_value(use.lr)))

            if countdown == - setcountdown:
                print(name + '    Training stopped at ' +
str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')) +
                          '       loss :' + str("%11.6f"% (loss_result[-1])))
                print('     Epoch ' + str("%5d"%(i+1)) + '     <'    +
str("%8.6f"%keras.backend.get_value(use.lr)))

                break

    print(name + '    Training Finished at ' +
str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
    logfile.close()
    pyplot.subplot(211)
    pyplot.plot(loss_result, label='train mse')
#     pyplot.plot(v_loss_result, label='test mse')
    pyplot.legend()
    pyplot.subplot(212)
```

```
        pyplot.plot(mae_result, label='train mae')
#         pyplot.plot(v_mae_result, label='test mae')
        pyplot.legend()
        pyplot.show()

        return model, pd.DataFrame({'loss_result':loss_result, 'mae_result':mae_result})

#%%
#Save model
#model.load_weights('D:/temp/model_weights.h5')
#trainingpath = pd.DataFrame([loss_result, v_loss_result, [min(v_loss_result)]],
index=['loss_result','v_loss_result','min_v'])
#trainingpath.to_csv('D:/archive/'+folder+'/trainingpath.csv')
#
#model.save('D:/temp/'+ str(datetime.date.today())+'_model.h5')
#model.save_weights('D:/temp/model_weights.h5')
#
#model.save('D:/archive/'+folder+'/'+ str(datetime.date.today())+'_model.h5')
#model.save_weights('D:/archive/'+folder+'/model_weights.h5')
#Save result

#%%

#读取网格搜索用的参数网格
#modelspace = pd.read_table('D:/temp/modelspace.txt', header=None,
                                        #index_col=[0], prefix='L')
#modelspace.to_csv('D:/archive/'+folder+'/modelspace.txt', sep='\t', header=False)

#%%
#store = pd.HDFStore('D:/archive/'+folder+'/store.h5') #保存 h5 文件，方便日后用 python
处#理
#这是程序的主循环，创建模型，训练模型，保存结果，都在这个循环里，每次循环是
网#格搜索空间里的一个节点
#for name in modelspace.index:
    # print(name + str(datetime.datetime.now().strftime('    %Y-%m-%d %H:%M')))
model = buildmodel(48,32,24)
#       model = Sequential()
#       #model.add(TimeDistributed(Dense(24, activation=ReLU),
input_shape=(timestepstoputin, 24)))
#       model.add(CuDNNGRU(12, input_shape=(10, 24), return_sequences = True, stateful =
False
#                                       ))
##       model.add(Activation('elu'))
##       model.add(Dropout(0.4, noise_shape=None, seed=None))
#       model.add(CuDNNGRU(12, return_sequences=True))
##       model.add(Activation(relu))
##       model.add(Dropout(0.4, noise_shape=(None, 1, 12), seed=None))
#       model.add(CuDNNGRU(12, return_sequences=True))
##       model.add(Activation('elu'))
##       model.add(Dropout(0.4, noise_shape=(None, 1, 12), seed=None))
```

```
#       #model.add(CuDNNLSTM(1, return_sequences=True, stateful=False))
##        model.add(Activation('tanh'))
#       model.add(TimeDistributed(Dense(1)))
##        model.add(Activation('elu'))
##        model.add(Activation(relu))
#       #model.add(Activation('tanh'))
#       model.compile(loss='mse', optimizer=use, metrics=['mae'])
#       model.summary()
#       model, localresult = finaltrain(data_nor, model, name)
name = 'M1'
model, localresult = training(data_nor, model, name)
      #store[name] = localresult
predicting(data_test, model, name)
      #model.save('D:/archive/'+folder+'/'+ str(datetime.date.today()) + name +'_model.h5')
      #plot_model(model, show_shapes=True, show_layer_names=False,
to_file='D:/archive/'+folder+'/' + name +' model.png')
#demoshow(model)
#store.close()

#%%
#name = modelspace.index[0]
#model =
buildmodel(modelspace.loc[name][0],modelspace.loc[name][1],modelspace.loc[name][2])
#model, localresult = finaltrain(data_nor, model, name)
#predicting(data_test, model)
#
#model.save('D:/archive/'+folder+'/'+ str(datetime.date.today()) + name +'_model.h5')
#demoshow(model)

#%%
#name = 'M1120 180'
#
#model= keras.models.load_model('D:/archive/1120-3/2018-11-20M1 180_model.h5')
#predicting(data_test, model, name)
```

附录 2 贝叶斯调参代码

```
from hyperopt import fmin,tpe,hp,STATUS_OK,Trials
import math
import hyperopt.pyll.stochastic
from matplotlib import pyplot as plt

fspace = {'l_rate':hp.uniform('l_rate',0.0005,0.0015)}

def f(params):
    model = buildmodel(params)


    model, localresult = training(data_nor, model, name)
    acc = localresult.loc[localresult.shape[0]-1,'mae_result']

    return {'loss': acc, 'status': STATUS_OK}

trials = Trials()
best = fmin(
        fn = f,
        space = fspace,
        algo = tpe.suggest,
        max_evals = 50,
        trials = trials
        )

print('best:',best)

for trial in trials.trials:
    print(trial)
```

附录 3 PCA 代码

```
# -*- coding: utf-8 -*-
"""
Created on Tue Oct  8 11:47:44 2019

@author: hushuli
"""

import pandas as pd
from sklearn.decomposition import PCA
import numpy as np
import pandas as pd
data = pd.read_table('train.txt', delim_whitespace=True)
data = pd.read_table('train.txt', delim_whitespace=True, header=None, index_col=[0,1])
data_1 = data.loc[1]

estimator = PCA(n_components=7)
estimator1 = PCA(n_components=1)

data_pca = estimator.fit_transform(data_1)

M = np.array([[1,2],[2,4],[5,6]])
N = estimator1.fit_transform(M)

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from datetime import datetime

values = data_pca

scaler = MinMaxScaler(feature_range=(0,1))
values = scaler.fit_transform(values)

groups = [0,1,2,3,4,5,6]

i = 1
plt.figure()
for group in groups:
	plt.subplot(len(groups),1,i)
	plt.plot(values[:,group])
#	  plt.title(dataset.columns[group], y =0.5, loc='right')
	plt.title(group, y =0.5, loc='right')
```

```
        i = i+1
plt.show()
```

附录 4 Conv-LSTM 模型部分代码

```
def buildmodel(ly1u, ly2u, ly3u):
    model = Sequential()
    model.add(ConvLSTM2D(filters=5, kernel_size=(1,6),
                         input_shape=(1, timestepstoputin, 24, 1), return_sequences=True))
    model.add(BatchNormalization())

    model.add(ConvLSTM2D(filters=5, kernel_size=(1,6),
                         return_sequences=True))
    model.add(BatchNormalization())

    model.add(ConvLSTM2D(filters=5, kernel_size=(1,6),
                         return_sequences=True))
    model.add(BatchNormalization())

    model.add(ConvLSTM2D(filters=5, kernel_size=(1,9),
                         return_sequences=True))
    model.add(BatchNormalization())

    model.add(TimeDistributed(Dense(1)))
    #model.compile(loss='binary_crossentropy', optimizer='adadelta')
    model.compile(loss='mse', optimizer=use, metrics=['mae'])
    model.summary()
    return model
```

攻读硕士学位期间取得的学术成果

（1）EI 论文《Research on prognosis by RNN deep learning method》

致谢

时光总是匆忙，两年半的研究生生涯转瞬即逝，非常高兴能在北航生活学习，这段求学经历，收获良多，开阔了视野，对国家的航空航天事业有了深入的了解，认识了很多优秀的老师和同学，我也对有了重新的审视和定位。由衷的感谢北航，圆了我 985 名校的梦想，感谢学校里支持鼓励我的人们。

首先感谢我的导师——张叔农老师。本论文的相关工作均是在张老师的指导下完成的。张老师鼓励她的每个学生充分发挥主观能动性，教我们如何将理论应用于实践，解决实际问题。在论文和项目的实施过程中，每当遇见问题，张老师总会给出耐心的指导，并且非常乐意同学生交流心得，教我们静下心来，冷静的分析和解决当前的问题。从开题到论文完成的每一个环节，张老师都付出了大量的精力。再次向表达张老师深切的祝福和感谢。

感谢实验室的唐亮，门伟阳，俞小勇，石文，李占领，沈德峰师兄的指导与关心，感谢王泽，袁金帅，杨雪松、李游、杨松、马启超师弟，徐幸师妹一起营造的和谐温馨的实验室氛围，感谢蓝煜东同学的算法指导，感谢刘金星，李桥，霍昊得室友以及杜小松，一起毕业走向社会，希望以后的人生大家一帆风顺。

感恩我的父亲，母亲，姥姥，舅舅，一直在默默的支持我，无私的付出， 希望家人都健康快乐。

最后，感谢可靠性与系统工程学院各位授业解惑的老师，感谢每一位审阅这篇论文的老师和专家，谢谢您的评论和建议。

胡树立

2019 年 11 月于北京