**同元AI系列工具箱应用案例--涡轮发动机剩余寿命预测**

作者：曹宇宇

# 背景介绍

涡轮发动机的工作状态直接影响飞机的性能。由于飞机需要频繁地完成起飞、巡航、着陆等任务，在飞行过程中如果发动机发生故障，不管是对人身安全方面的危害，还是经济方面的损失，都是无法估量的。因此，涡轮发动机的剩余使用寿命(remaining useful life，RUL)对于规避永久性损坏，降低运营和维护成本，提高运行安全具有重要意义。

RUL作为预测与健康管理(Prognostics and health management，PHM)技术的两大组成部分之一，通常用于描述当前时刻与失效时刻之间的时间间隔，可定义为

式中：T表示设备的失效时刻，t表示当前时刻。RUL预测对维修决策和备件订购具有指导性价值，长期以来被当作PHM技术的基础和核心。主流的RUL方法主要分为三类：基于失效机理的分析方法、数据驱动的方法以及失效机理分析和数据驱动模型相融合的方法。基于失效机理的分析方法和两者相融合的方法都需要通过数学物理模型来描述设备的失效机理，建模过程较为复杂。数据驱动的方法则不太依赖设备的失效机理，而机器学习的出现则加速了PHM这一领域的成熟。因此，基于机器学习的剩余寿命预测越来越得到认可，尤其是在复杂工程中得到越来越多的应用。

本文利用同元深度学习工具箱(MWORKS.Syslab TyDeepLearning ToolBox)，并基于深度学习中的卷积神经网络(Convolutional Neural Network，CNN)和长短时记忆网络(Long Short Term Memory Network, LSTM)实现了对涡轮发动机RUL的预测，取得了较好的结果。

MWORKS.Syslab是同元软控全新推出的新一代科学计算环境，基于高性能科学计算语言Julia提供交互式编程环境的完备功能。在实际工程中，通过将算法应用到实际设备中，构建数据模型来描述系统或部件的状态，能够极大地节约人力物力，RUL预测便是Syslab的应用领域之一。随着同元软控与华为的合作进一步深入，我们以昇思MindSpore为框架底座，打造了同元 AI系列工具箱，已于2023年1月8号正式对外发布。

下面将基于深度学习工具箱进行涡轮发动机剩余寿命预测的算法设计和实现。

# 数据集介绍

本文使用NASA涡轮风扇发动机退化仿真数据集。训练数据包含100台发动机的从完整运行至故障的仿真时间序列数据。测试数据包含100个不完整序列，每个序列的末尾为相应的剩余使用寿命值。旨在根据发动机中各种传感器的时间序列数据来预测发动机的剩余使用寿命（预测性维护，以周期为单位度量）。

从<https://www.nasa.gov/intelligent-systems-division>下载并解压缩涡轮风扇发动机退化仿真数据集。

涡轮风扇发动机退化仿真数据集的每个时间序列表示一个发动机。每台发动机启动时的初始磨损程度未知。发动机在每个时间序列开始时运转正常，在到达序列中的某一时刻时出现故障。在训练集中，故障的规模不断增大，直到出现系统故障。

数据包含 26 列以空格分隔的数值。每一行是在一个运转周期中截取的数据快照，每一列代表一个不同的变量。这些列分别对应于以下数据：

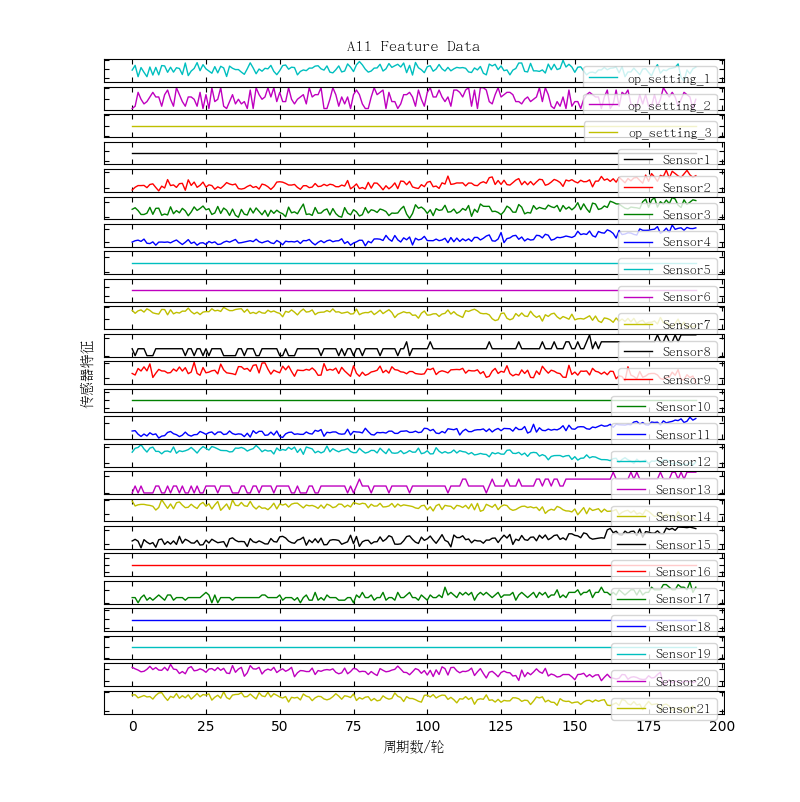
- 第 1 列 - 单元编号

- 第 2 列 - 周期时间

- 第 3-5 列 - 操作设置

- 第 6-26 列 - 传感器测量值 1-21

将3-26列作为特征数据，绘制展示第一台发动机的特征数据。如下图所示：



△ 全量特征曲线图

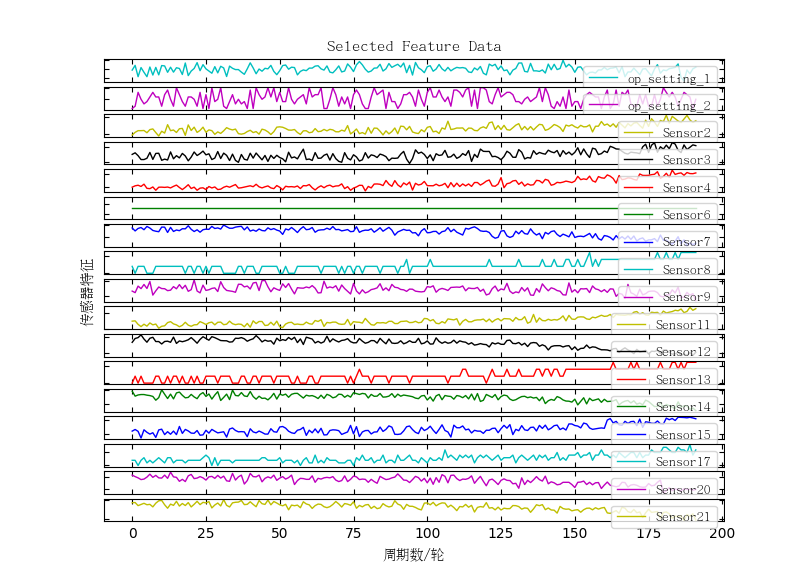
# 数据预处理

## 删除具有常量值的特征

所有时间步都保持不变的特征可能无法提取对训练结果有用的信息。找到最小值和最大值相同的数据行，然后删除这些行。

删除的特征包括：["op\_setting\_3", Sensor1", "Sensor5", "Sensor10", "Sensor16", "Sensor18", "Sensor19"]，这与特征曲线图中所绘制的情形相同。序列中剩余的特征数为17个。

将第一台发动机剩余的特征绘制成特征曲线图，如下图所示：



△ 特征曲线图

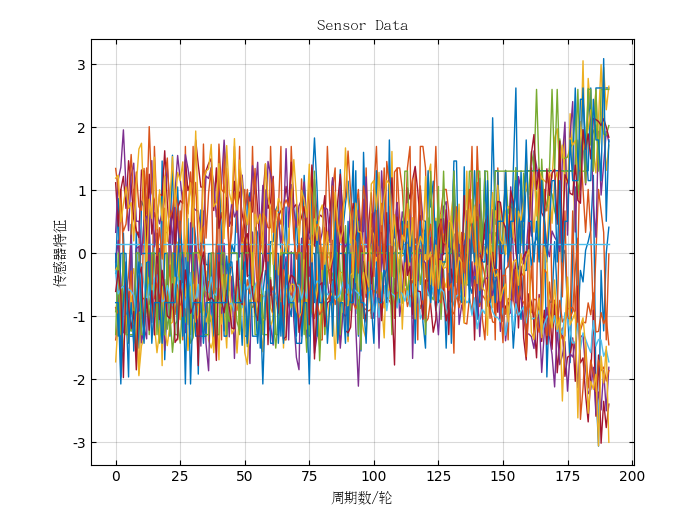
## 标准化训练预测变量

将训练预测变量进行标准化。用待标准化数据减去均值，然后除以标准差，经过处理后数据符合标准正态分布，即均值为0，标准差为1。

标准化方程为：

式中： 代表特征列的平均值，代表特征列的标准差。

对第一台发动机的17维传感器参数进行标准化处理后如下图所示：



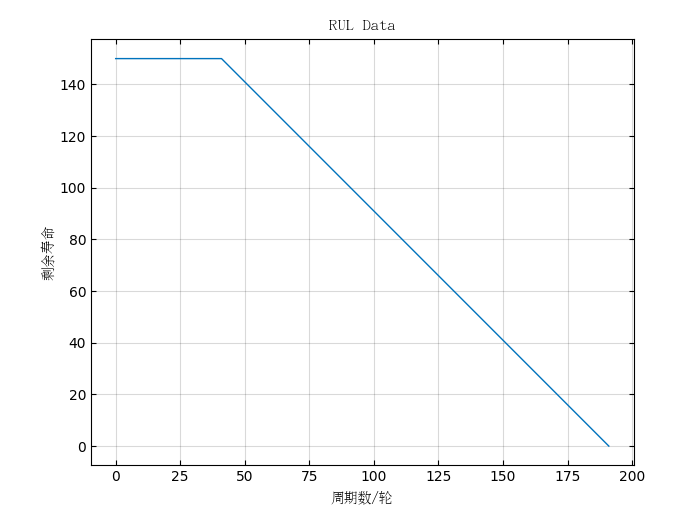
△ 标准化后曲线

## 剩余寿命估计目标函数

机械设备工作初期设备磨损较小，工作处于稳定状态。经过长时间的磨损，机械设备性能逐渐降低，当磨损达到临界值，设备的使用寿命将逐渐下降。因此，使用分段目标函数能够更加准确的预测剩余寿命的发展趋势。

在本案例中，对于一个完整运行寿命周期为200的实验，设其最初运行时为最大使用寿命150，运行一段时间后，电机寿命开始逐渐下降，呈线性衰减，实验结束，剩余使用寿命衰减为0。

下图显示了第一台发动机的剩余寿命趋势图：



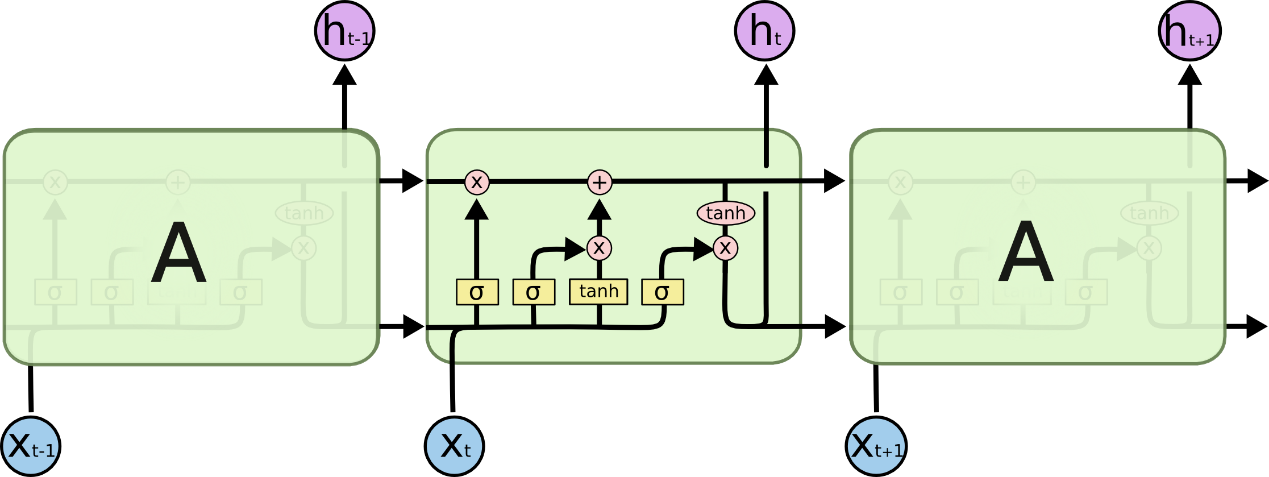
△ 剩余寿命趋势图

# 模型训练

本文使用深度学习工具箱中的长短时记忆网络、卷积神经网络构建模型，进而比较各种模型的优劣。

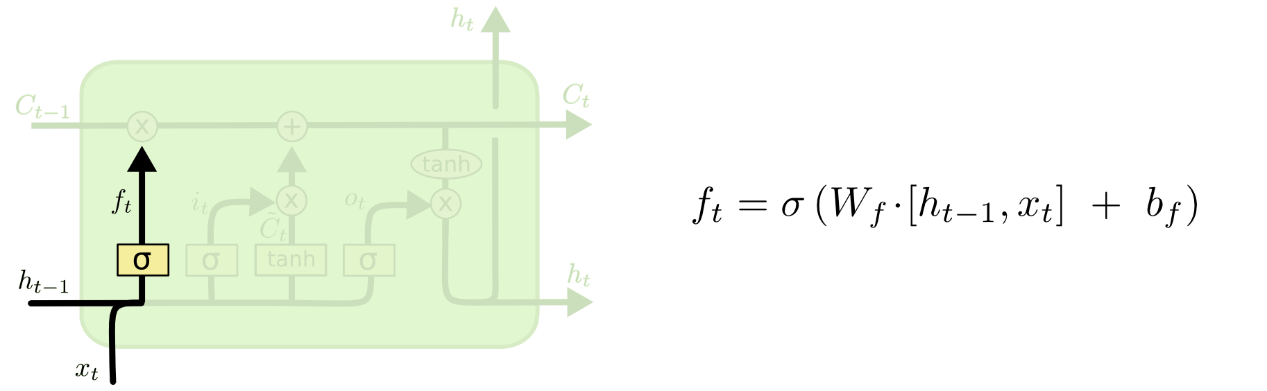
## 基于LSTM的剩余寿命预测

LSTM，是一种改进之后的循环神经网络，可以解决循环神经网络中无法处理长距离的依赖的问题。其结构图如下所示：



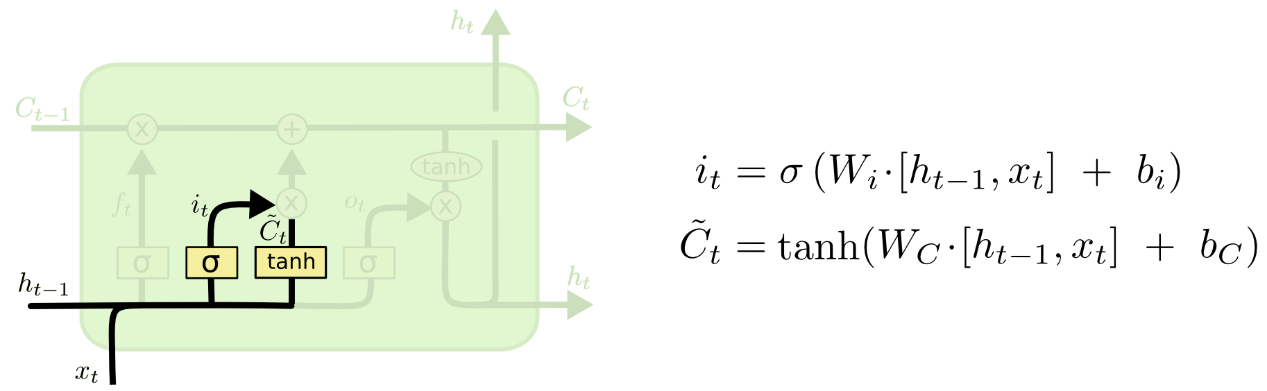
△ LSTM结构图

可以看到，LSTM主要有三个门组成，即遗忘门、输入门和输出门。



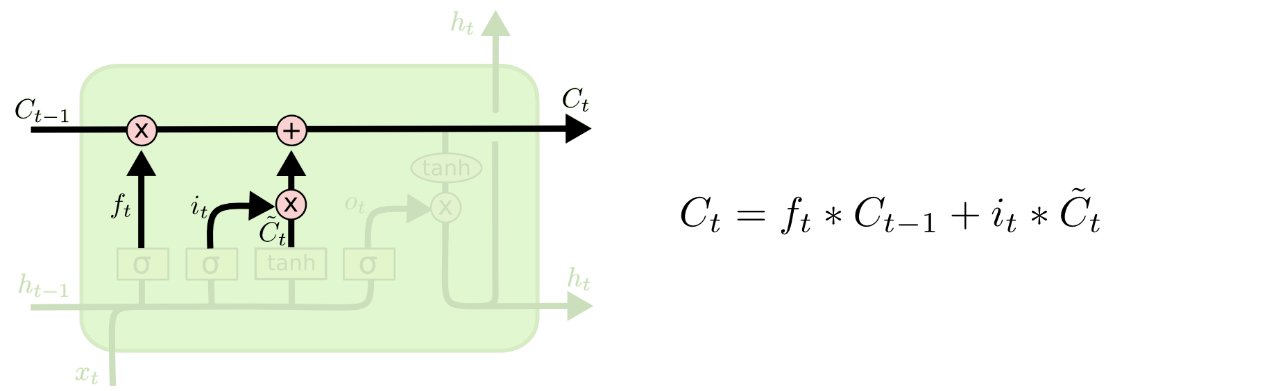
△ LSTM遗忘门

该门会读取隐层状态 的信息和t时刻输入向量的信息，输出一个在 0 到 1 之间的数值，并传递给携带过去信息的细胞状态。1 表示“完全保留”，0 表示“完全舍弃”。



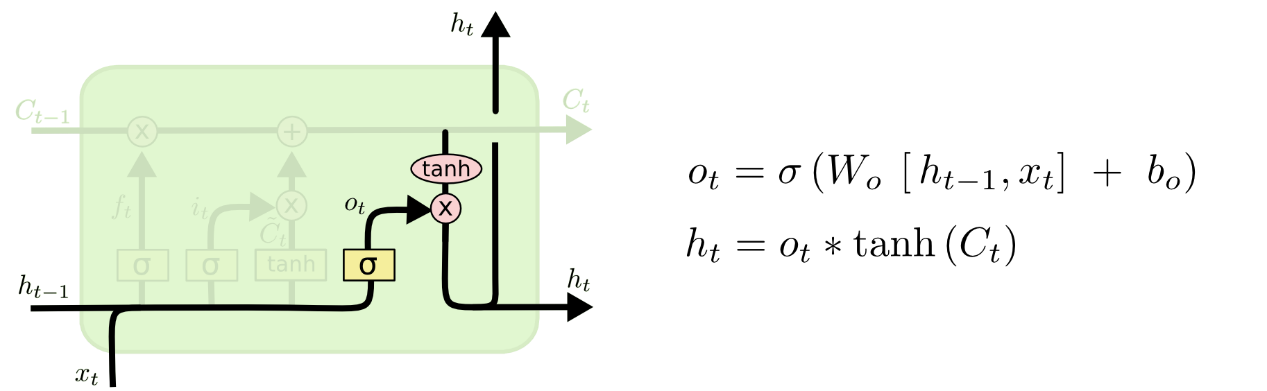
△ LSTM输入门(一)

下一步是确定什么样的新信息被存放在细胞状态中。这里包含两个部分。首先，sigmoid层决定什么值将要被更新。然后，tanh 层创建一个新的候选值向量，这个可以看作是新的输入带来的信息。



△ LSTM输入门(二)

下面就是对的更新，公式左边是前面遗忘门给出的，这个值乘，表示过去的信息有选择地遗忘或保留。右边是用前面输入门给出的乘新的候选值向量，表示新的信息有选择地遗忘或保留。最后，再把这两部分信息加起来，就是新的状态了。



△ LSTM输出门

最终，输出将基于细胞状态，但也是一个过滤后的版本。首先，运行一个 sigmoid 层来确定细胞状态的哪个部分将输出出去。接着，把细胞状态通过 tanh 进行处理（得到一个在-1到 1之间的值）并将它和 sigmoid 门的输出相乘，最后仅仅会输出我们确定输出的那部分。

### 定义网络架构

创建一个 LSTM 网络，该网络包含一个具有 200个隐藏单元的 LSTM 层，然后是一个输出大小为 50 的全连接层和一个丢弃概率为 0.5 的丢弃层，最后连接一个输出大小为1的全连接层。

#Julia Code

numFeatures = 17

numHiddenUnits = 200

layers = SequentialCell([

lstmLayer(numFeatures, numHiddenUnits; NumLayers=1, Batch\_First=true),

flattenLayer(),

fullyConnectedLayer(numHiddenUnits \* sequence\_length, 100),

reluLayer(),

dropoutLayer(0.5),

fullyConnectedLayer(100, 1),

])

### 训练网络

使用 trainNetwork 训练网络。

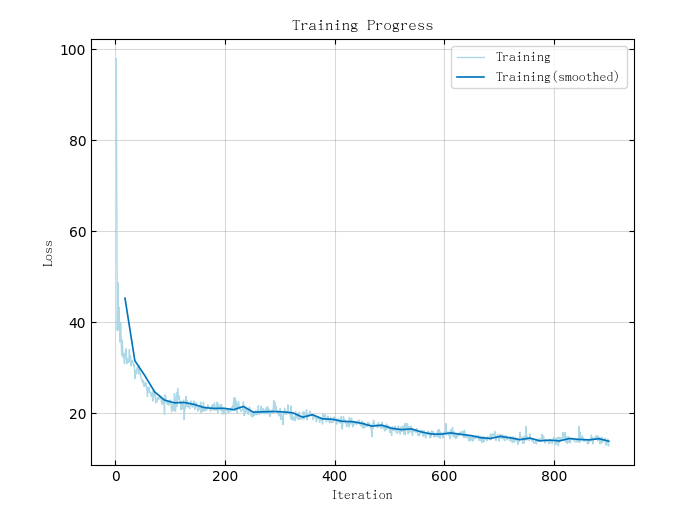
#Julia Code

options = trainingOptions("RMSELoss", "Adam", "MSE", 1024, 50, 0.01; Plots=true)

net = trainNetwork(XTrain, YTrain, layers, options)

其中：trainingOptions为训练所需参数设置，损失函数为RMSELoss，优化器为Adam，评估指标为MSE，BatchSize为1024，Epoch为50，学习率为0.01。

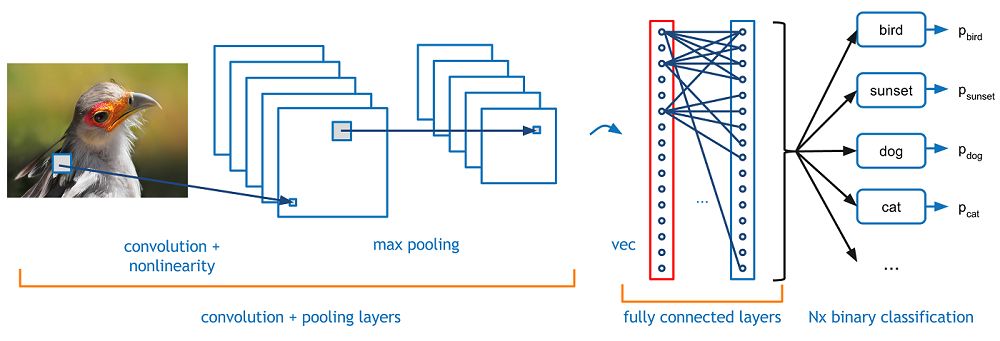
训练结果如下图所示：



△ LSTM Loss曲线图

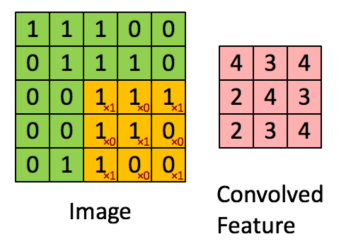
## 基于CNN的剩余寿命预测

CNN，它是属于前馈神经网络的一种，其特点是每层的神经元节点只响应前一层局部区域范围内的神经元。一个深度卷积神经网络模型，一般由若干卷积层叠加若干全连接层组成，中间包含各种的卷积操作、池化操作。其结构图如下所示：



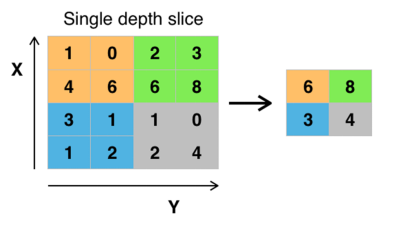
△ CNN结构图

可以看到，CNN主要有三个层组成(从左至右)，即卷积层、池化层和全连接层。



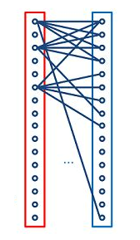
△ CNN卷积层

卷积层中最重要的概念是卷积核(图左橙色部分)，卷积核可以理解为是一种特征，将输入和卷积核相乘得到的结果就是输入在这个特征上的投影，这个投影可以称之为特征图。



△ CNN池化层

常见的池化有最大值池化(如图所示)和平均池化两种，顾名思义就是在池化窗口内计算最大值和平均值作为池化结果。池化层的作用主要体现在：特征不变性、特征降维和减少过拟合。



△ 全连接层

全连接层一般位于整个神经网络的最后，负责将卷积输出的二维特征图（矩阵）转换成一维向量。

对于文本分类和时间序列预测等简单任务，小型的一维卷积神经网络可以替代RNN，而且速度更快。

### 定义网络架构

创建一个 CNN 网络，该网络包含一个具有 5个一维卷积层，然后是一个展平层，接着是一个输出大小为 100 的全连接层和一个丢弃概率为 0.5 的丢弃层，最后连接一个输出大小为1的全连接层。

#Julia Code

layers = SequentialCell([

convolution1dLayer(17, 32, 5),

reluLayer(),

convolution1dLayer(32, 64, 7),

reluLayer(),

convolution1dLayer(64, 128, 11),

reluLayer(),

convolution1dLayer(128, 256, 13),

reluLayer(),

convolution1dLayer(256, 512, 15),

reluLayer(),

flattenLayer(),

fullyConnectedLayer(512 \* sequence\_length, 100),

reluLayer(),

dropoutLayer(0.5),

fullyConnectedLayer(100, 1),

])

### 训练网络

使用 trainNetwork 训练网络。

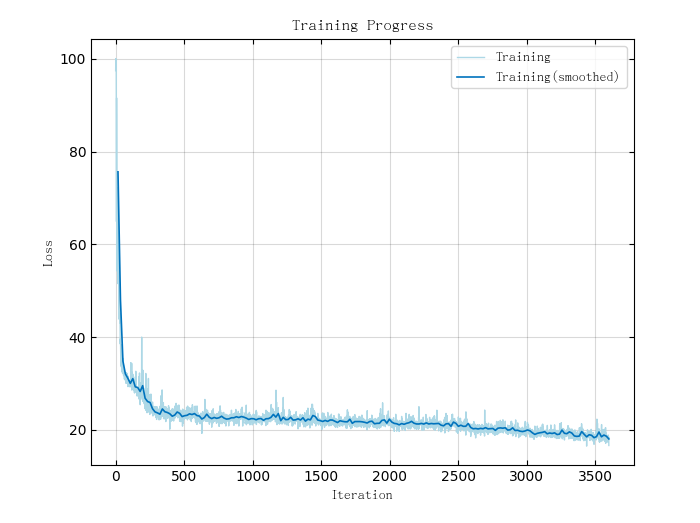
#Julia Code

options = trainingOptions("RMSELoss", "Adam", "MSE", 512, 100, 0.001; Plots=true)

net = trainNetwork(XTrain, YTrain, layers, options)

其中：trainingOptions为训练所需参数设置，损失函数为RMSELoss，优化器为Adam，评估指标为MSE，BatchSize为512，Epoch为100，学习率为0.001。

训练结果如下图所示：



△ CNN Loss曲线图

# 模型验证与评估

## 模型验证

本文使用此数据集自带的测试数据进行验证，该数据集同样进行上述数据预处理，使用 predict 对测试数据进行预测。

#Julia Code

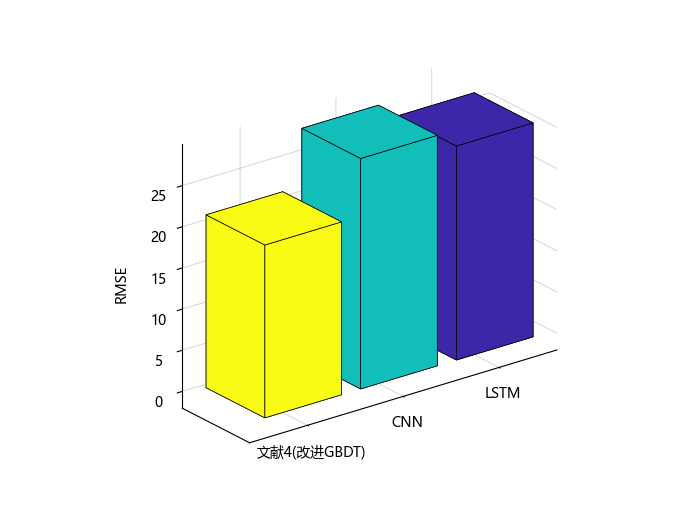
TyDeepLearning.predict(net, XTest)

计算真实值与预测值的均方根误差：

式中：代表预测值，代表真实值。

在柱状图中可视化预测误差，如下图所示：

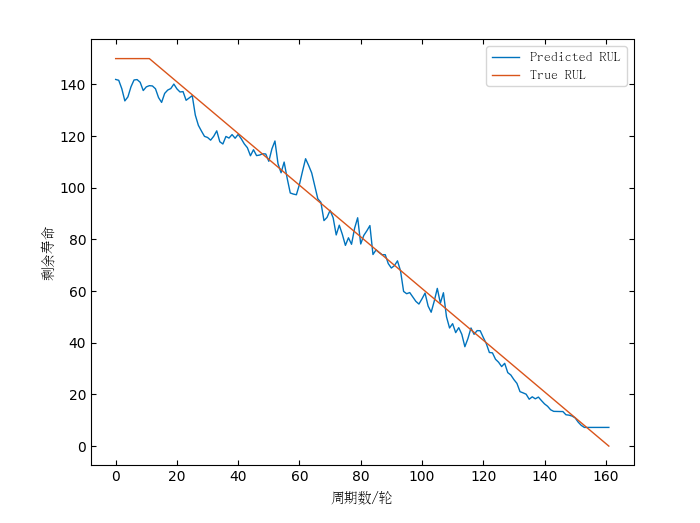
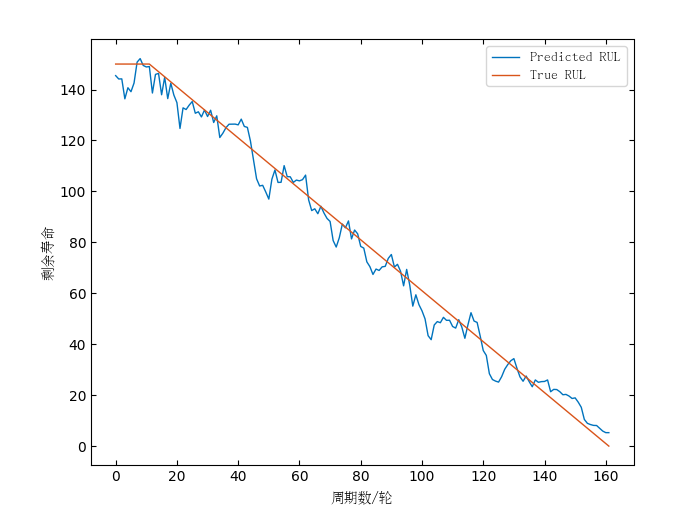
|  |  |
| --- | --- |
| 算法 | RMSE |
| CNN | 26.15 |
| LSTM | 28.04 |
| 改进GBDT | 20.88 |



△ 不同算法RMSE对比图(左)和对比表(右)

## 结果评估

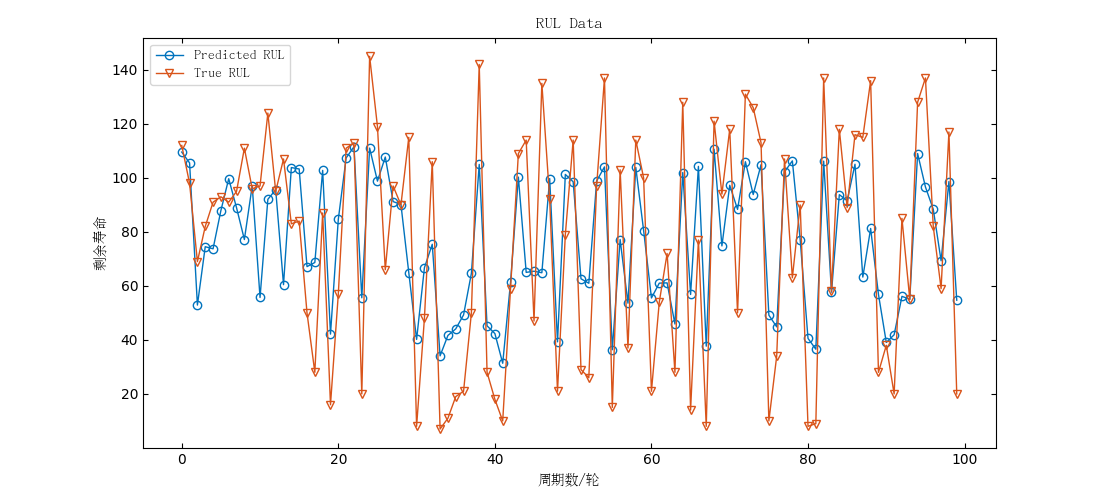
为了对训练效果进行评估，下图展示了第一台发动机在训练集中真实值与预测值的剩余寿命预测对比：

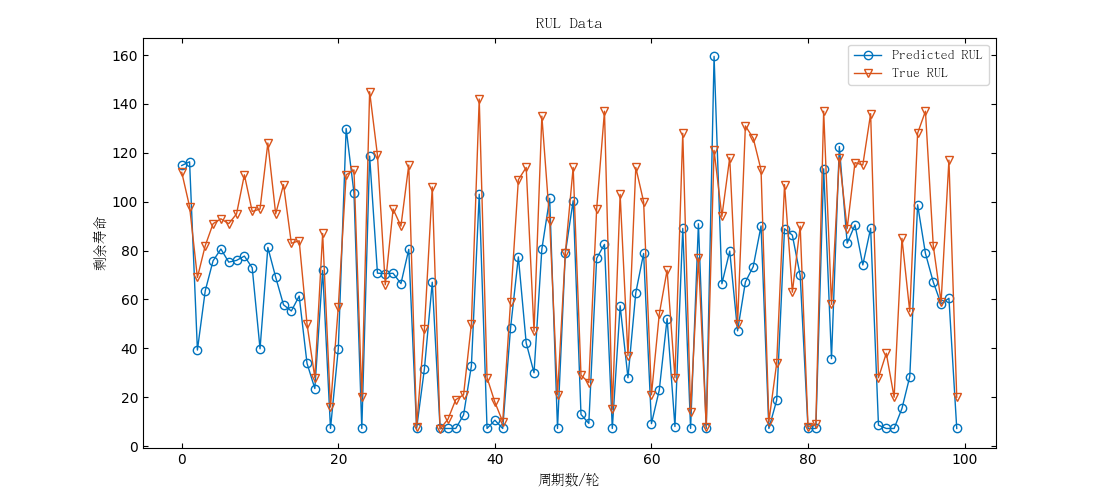


△ LSTM(左)与CNN(右)预测对比图

从图中可以看到，在训练集上，使用LSTM要比CNN拟合效果更好，更能贴近真实的剩余寿命值。

为了对测试集中的结果进行评估，下图展示了第一台发动机在测试集中真实值与预测值的剩余寿命预测对比：





△ LSTM(上)与CNN(下)预测对比图

综上，不论是LSTM还是CNN，其预测的结果与真实值之间符合偏差要求，并且趋势相同，可以较为真实的反应实际的场景。

# 模型保存与加载

深度学习工具箱提供了模型保存与加载的函数，可以将模型快速进行保存，支持函数重复调用，模型部署与推理，用户可便捷地将训练好的模型应用到实际场景中，真正实现涡轮发动机剩余寿命预警。

#Julia Code

#保存模型

file = "/home/tongyuan/example/RUL.ckpt"

net = save(layers, file)

#加载模型

file = "/home/tongyuan/example/RUL.ckpt"

net = load(file; net= layers)

# 总结

本文主要基于同元深度学习工具箱(TyDeepLearning ToolBox)设计了涡轮发动机剩余寿命预测算法，并采用卷积神经网络和循环神经网络训练模型，分析了两种情况下训练的模型对寿命预测的效果，筛选出了合适的模型，并能够支持实际的应用场景。

同元深度学习工具箱内含丰富的函数库、应用案例及完善的帮助文档。用户可基于深度学习工具箱，采用Julia语言构建多种神经网络，并进行训练计算。。目前，深度学习工具箱在图像识别、语音识别和故障预测等领域已实现多项应用，取得较好的效果。

未来，我们将进一步扩充深度学习工具箱函数库、基于Julia语言实现多种深度学习底层框架自由选择、完善GPU运算功能及分布式并行计算能力、探索代码生成逻辑、在此基础上实现与MWORKS.Sysplorer相结合的机理数据融合建模。最终，构建多硬件平台、多框架、快速运算、功能完备的新一代AI系列工具箱。

# 代码附录

using TyDeepLearning

using DataFrames

using TyPlot

dir = "data/RUL/"

# 训练集数据处理

path1 = dir \* "train\_FD001.csv"

train\_FD001 = CSV.read(path1, DataFrame; header=false)

# 寻找每个单元的最大cycle

train\_count = zeros(Int64, (100))

for i in range(1, size(train\_FD001)[1])

for j in range(1, 100)

if train\_FD001[i, 1] == j

train\_count[j] = train\_count[j] + 1

else

continue

end

end

end

RUL = zeros(Int64, (size(train\_FD001)[1]))

for i in range(1, size(train\_FD001)[1])

for j in range(1, 100)

if train\_FD001[i, 1] == j

RUL[i] = train\_count[j] - train\_FD001[i, 2]

else

continue

end

end

end

# 删除某些在所有步长中保持不变的特征

select!(

train\_FD001,

Not([:Column5, :Column6, :Column10, :Column15, :Column21, :Column23, :Column24]),

)

# 添加一列名为RUL

train\_FD001 = insertcols!(train\_FD001, ncol(train\_FD001) + 1, :RUL => RUL)

# 标准化

feats = Array(train\_FD001[!, 3:19])

data\_mean = mean(feats; dims=1)

data\_std = std(feats; dims=1)

feats\_norm = (feats .- data\_mean) ./ data\_std

# 滑动窗口为31

sequence\_length = 31

function gen\_sequence(data\_array, data\_label, seq\_length)

num\_elements = size(data\_array)[1]

label = data\_label[seq\_length:num\_elements]

data = reshape(data\_array[1:seq\_length, :], (1, seq\_length, 17))

dict = zip(range(2, num\_elements - seq\_length + 1), range(seq\_length + 1, num\_elements))

for (start, stop) in dict

data = vcat(data, reshape(data\_array[start:stop, :], (1, seq\_length, 17)))

end

return data, label

end

count\_sum = zeros(Int64, (101))

for i in range(1, 100)

count\_sum[i + 1] = train\_count[i] + count\_sum[i]

end

data\_label = RUL[1:192]

data\_array = feats\_norm[1:192, :]

data, label = gen\_sequence(data\_array, data\_label, sequence\_length)

for j in range(2, 100)

data\_array = feats\_norm[(count\_sum[j] + 1):count\_sum[j + 1], :]

data\_label = RUL[(count\_sum[j] + 1):count\_sum[j + 1]]

data\_id, data\_id\_label = gen\_sequence(data\_array, data\_label, sequence\_length)

data = cat(data, data\_id; dims=1)

label = cat(label, data\_id\_label; dims=1)

end

# 裁剪响应RUL阈值为150

RUL\_Threshold = 150

for i in range(1, size(label)[1])

if label[i] > 150

label[i] = 150

end

end

XTrain = permutedims(data, (1, 2, 3))

YTrain = label

# LSTM网络构建

numFeatures = 17

numHiddenUnits = 200

layers = SequentialCell([

lstmLayer(numFeatures, numHiddenUnits; NumLayers=1, Batch\_First=true),

flattenLayer(),

fullyConnectedLayer(numHiddenUnits \* sequence\_length, 50),

reluLayer(),

dropoutLayer(0.5),

fullyConnectedLayer(50, 1),

])

options = trainingOptions("RMSELoss", "Adam", "MSE", 1024, 50, 0.01; Plots=true)

net = trainNetwork(XTrain, YTrain, layers, options)

# CNN网络构建

layers = SequentialCell([

convolution1dLayer(17, 32, 5),

reluLayer(),

convolution1dLayer(32, 64, 7),

reluLayer(),

convolution1dLayer(64, 128, 11),

reluLayer(),

convolution1dLayer(128, 256, 13),

reluLayer(),

convolution1dLayer(256, 512, 15),

reluLayer(),

flattenLayer(),

fullyConnectedLayer(512 \* sequence\_length, 100),

reluLayer(),

dropoutLayer(0.5),

fullyConnectedLayer(100, 1),

])

options = trainingOptions("RMSELoss", "Adam", "MSE", 512, 100, 0.001; Plots=true)

net = trainNetwork(XTrain, YTrain, layers, options)

# 测试集数据处理

path2 = dir \* "test\_FD001.csv"

path3 = dir \* "RUL\_FD001.csv"

test\_FD001 = CSV.read(path2, DataFrame; header=false)

RUL\_FD001 = CSV.read(path3, DataFrame; header=false)

select!(

test\_FD001,

Not([:Column5, :Column6, :Column10, :Column15, :Column21, :Column23, :Column24]),

)

# 寻找每个单元的最大cycle

test\_count = zeros(Int64, (100))

for i in range(1, size(test\_FD001)[1])

for j in range(1, 100)

if test\_FD001[i, 1] == j

test\_count[j] = test\_count[j] + 1

else

continue

end

end

end

test\_count\_sum = zeros(Int64, (101))

for i in range(1, 100)

test\_count\_sum[i + 1] = test\_count[i] + test\_count\_sum[i]

end

test\_RUL = zeros(Int64, (size(test\_FD001)[1]))

for i in range(1, size(test\_FD001)[1])

for j in range(1, 100)

if test\_FD001[i, 1] == j

test\_RUL[i] = test\_count[j] - test\_FD001[i, 2] + RUL\_FD001[j, 1]

else

continue

end

end

end

# 标准化处理

data\_test = Array(test\_FD001[!, 3:19])

data\_mean = mean(data\_test; dims=1)

data\_std = std(data\_test; dims=1)

test\_norm = (data\_test .- data\_mean) ./ data\_std

test\_data\_array = test\_norm[(test\_count\_sum[1] + 1):test\_count\_sum[1 + 1], :]

test\_data = reshape(

test\_data\_array[(test\_count[1] - sequence\_length + 1):end, :], (1, sequence\_length, 17)

)

for j in range(2, 100)

test\_data\_array = test\_norm[(test\_count\_sum[j] + 1):test\_count\_sum[j + 1], :]

datadata = test\_data\_array[(test\_count[j] - sequence\_length + 1):end, :]

data\_reshape = reshape(datadata, (1, sequence\_length, 17))

test\_data = cat(test\_data, data\_reshape; dims=1)

end

# 裁剪响应RUL阈值为150

YTest = Array(RUL\_FD001)

RUL\_Threshold = 150

for i in range(1, size(YTest)[1])

if YTest[i] > 150

YTest[i] = 150

end

end

XTest = permutedims(test\_data, (1, 3, 2))

Y = TyDeepLearning.predict(net, XTest)

error = sqrt(mse(YTest, Y))

#保存模型

file = "/home/tongyuan/example/RUL.ckpt"

net = save(layers, file)

#加载模型

file = "/home/tongyuan/example/RUL.ckpt"

net = load(file; net= layers)

# 参考文献

1. 裴洪，胡昌华，司小胜，等. 基于机器学习的设备剩余寿命预测方法综述 [J]. 机械工程学报，2019，55(08).
2. 刘国建，杜冬，邢苗英，翟羽佳.基于机器学习的航空发动机剩余寿命预测综述[J].电子技术, 2023, 52(01): 52-53.
3. 杨秋玉，阮江军，黄道春，等．基于VMD-Hilbert边际谱能量熵和SVM的高压断路器机械故障诊断［J］．电机与控制学报，2020，24(3):11．
4. 柳长源，何先平，于会越. 基于数据驱动的涡轮发动机剩余寿命预测[J].电机与控制学报，2021.
5. Saxena, Abhinav, Kai Goebel, Don Simon, and Neil Eklund."Damage propagation modeling for aircraft engine run-to-failure simulation."In Prognostics and Health Management, 2008.PHM 2008.International Conference on, pp. 1-9.IEEE, 2008.
6. Saxena, Abhinav, Kai Goebel."Turbofan Engine Degradation Simulation Data Set."NASA Ames Prognostics Data Repository https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/, NASA Ames Research Center, Moffett Field, CA