

**《数据挖掘》课程设计（考试）论文（报告）评分表**

|  |  |  |
| --- | --- | --- |
| **姓名：赵云飞** | **班级：应数12202班** | |
| **学号：2226010427** | **E-mail：1971135468@qq.com** | |
| **项目代码地址：** | | |
| **考核项目** | | **考核分数** |
| **数据采集工具的选择、数据来源可靠性及完整性（10分）** | |  |
| **数据清洗与集成方法正确性（15分）** | |  |
| **数据挖掘过程中描述性、探索性分析的准确性（20分）** | |  |
| **数据可视化分析、对比方法、主要结论(观点)创新性（20分）** | |  |
| **技术报告的表达流畅性（15分）** | |  |
| **技术报告的结构合理性（10分）** | |  |
| **可视化分析工具的恰当性、附录代码的规范性（10分）** | |  |
| **总成绩（总分100分）** | |  |

|  |
| --- |
| 评语： |

**教师签名： 日 期：**

基于RNN与LSTM的ETTh1‘油温’预测

姓名：赵云飞

学号：2226010427 E-mail：1971135468@qq.com

**摘 要 ：**

**目的** 随着电力需求的日益增长，电力系统的高效调度成为全球能源管理中的核心挑战。特别是电力变压器的油温预测，作为关键设备的健康监测指标之一，具有重要的实践意义。油温波动受到多种因素的影响，包括负载变化、环境温度、季节波动等，这使得精确的长期预测变得尤为困难。传统的预测方法无法应对复杂的时序特征和多因素影响，导致预测结果存在较大偏差。为此，本文提出基于深度神经网络的时间序列预测方法，旨在提高电力变压器油温的预测精度，从而为电力变压器的运行与维护决策提供科学支持，减少资源浪费并优化电力调度。

**方法** 本研究使用的电力变压器数据集ETDataset，包含了来自中国同一省份的两个不同区域（ETT-small-m1和ETT-small-m2）2年内每分钟记录的数据，涵盖了8维特征，包括目标变量“油温”以及多个外部负载因素。数据展示了明显的短周期模式、长周期趋势及不规则波动。在模型构建方面，本文分别采用了循环神经网络（RNN）和长短期记忆网络（LSTM）进行油温的时间序列预测。为了提高训练效率和模型稳定性，本文采用了mini-batch训练方式，并通过RandomizedSearchCV进行超参数优化，选择了适合的批大小（batch size=64）和训练轮次（epoch=100）。此外，为避免过拟合，本文引入了早停（Early Stopping）策略，当验证集的损失在若干轮内未出现明显下降时，自动停止训练。为进一步优化模型性能，采用了Adam优化器，以提升模型的收敛速度和自适应能力。模型性能的评估指标包括均方误差（MSE）和平均绝对误差（MAE），以量化预测值与实际值之间的差异。

**结果** 在RNN模型的训练过程中，损失值从初始的0.2051逐渐降低至0.0084，训练过程中最大损失的下降幅度出现在第一轮，降幅达到0.1557。最终，RNN模型的MSE和MAE分别为30.4445和5.5177，表明该模型能够有效捕捉短期周期特征，但在长周期和不规则模式的预测上存在一定的误差。与RNN模型相比，LSTM模型的表现更为优越。在LSTM的训练过程中，损失值从0.0340下降至0.0062，训练过程中的最大损失下降幅度为0.061，训练总轮次为28轮，训练过程在模型性能达到最佳后自动停止。最终，LSTM模型的MSE和MAE分别为25.7840和5.0778，展示了较RNN模型更高的精度和稳定性。  
**结论** 本文通过比较RNN和LSTM模型在电力变压器油温预测中的表现，验证了深度神经网络在处理复杂时间序列数据中的优势。LSTM模型相较于RNN模型，在精度、稳定性及泛化能力上均表现更佳，特别是在处理长期趋势和复杂周期模式时，LSTM模型具有显著的优势。这一结果表明，基于LSTM的油温预测方法能够有效减少预测误差，进而优化电力系统的调度和设备管理，具有重要的应用价值和实际意义。未来的研究可进一步探索模型的可解释性和应用场景，以提升电力系统的智能化水平。

**关键词**：电力变压器；时间序列数据；MSE与MAE；循环神经网络（RNN）；长短期记忆网络（LSTM）

**ETTh1 ‘Oil Temperature’ prediction based on RNN and LSTM**

作者

*1.226010427 2. 1971135468@qq.com*

**Abstract:**

**Objective** With the growing demand for electricity, efficient dispatch of power systems has become a core challenge in global energy management. In particular, the prediction of oil temperature of power transformers, as one of the health monitoring indicators of key equipment, has important practical significance. Oil temperature fluctuations are affected by many factors, including load changes, ambient temperature, seasonal fluctuations, etc., which makes accurate long-term predictions particularly difficult. Traditional prediction methods are unable to cope with complex time series characteristics and multi-factor influences, resulting in large deviations in prediction results. To this end, this paper proposes a time series prediction method based on deep neural networks, aiming to improve the prediction accuracy of power transformer oil temperature, thereby providing scientific support for the operation and maintenance decisions of power transformers, reducing resource waste and optimizing power dispatch.

**Method** The power transformer dataset ETDataset used in this study contains data recorded every minute for 2 years from two different regions (ETT-small-m1 and ETT-small-m2) in the same province of China, covering 8-dimensional features, including the target variable "oil temperature" and multiple external load factors. The data shows obvious short-term patterns, long-term trends and irregular fluctuations. In terms of model construction, this paper uses recurrent neural networks (RNN) and long short-term memory networks (LSTM) for time series prediction of oil temperature. In order to improve training efficiency and model stability, this paper adopts mini-batch training, and uses RandomizedSearchCV for hyperparameter optimization, selecting a suitable batch size (batch size=64) and training rounds (epoch=100). In addition, to avoid overfitting, this paper introduces an early stopping strategy, which automatically stops training when the loss of the validation set does not show a significant decrease within several rounds. To further optimize the model performance, the Adam optimizer is used to improve the convergence speed and adaptability of the model. Evaluation metrics of model performance include mean squared error (MSE) and mean absolute error (MAE) to quantify the difference between the predicted values and the actual values.

**Result** During the training of the RNN model, the loss value gradually decreased from the initial 0.2051 to 0.0084. The maximum loss decrease during the training process occurred in the first round, with a decrease of 0.1557. Finally, the MSE and MAE of the RNN model were 30.4445 and 5.5177, respectively, indicating that the model can effectively capture short-term periodic features, but there are certain errors in the prediction of long-term and irregular patterns. Compared with the RNN model, the LSTM model performs better. During the training of the LSTM, the loss value decreased from 0.0340 to 0.0062, and the maximum loss decrease during the training process was 0.061. The total number of training rounds was 28, and the training process automatically stopped after the model performance reached the best. Finally, the MSE and MAE of the LSTM model were 25.7840 and 5.0778, respectively, showing higher accuracy and stability than the RNN model.

**Conclusion** This article verifies the advantages of deep neural networks in processing complex time series data by comparing the performance of RNN and LSTM models in predicting power transformer oil temperature. Compared with the RNN model, the LSTM model performs better in terms of accuracy, stability and generalization ability. Especially when dealing with long-term trends and complex periodic patterns, the LSTM model has significant advantages. This result shows that the oil temperature prediction method based on LSTM can effectively reduce prediction errors, thereby optimizing the dispatching and equipment management of the power system, and has important application value and practical significance. Future research can further explore the interpretability and application scenarios of the model to improve the intelligence level of the power system.、

**Key words：**Power transformer; time series data; MSE and MAE; recurrent neural network (RNN); long short-term memory network (LSTM)

**0 引 言**

电力分配问是电网根据顺序变化的需求管理电力分配到不同用户区域。但要预测特定用户区域的未来需求是困难的，因为它随工作日、假日、季节、天气、温度等的不同因素变化而变化。现有预测方法不能适用于长期真实世界数据的高精度长期预测，并且任何错误的预测都可能产生严重的后果。因此当前没有一种有效的方法来预测未来的用电量，管理人员就不得不根据经验值做出决策，而经验值的阈值通常远高于实际需求。保守的策略导致不必要的电力和设备折旧浪费。值得注意的是，变压器的油温可以有效反映电力变压器的工况。最有效的策略之一是预测变压器的油温同时设法避免不必要的浪费。 为了解决这个问题，Zhou的团队与北京国网富达科技发展公司建立了一个平台并收集了2年的数据(Zhou等 ，2021)。我们用它来预测电力变压器的油温并研究电力变压器极限负载能力[1]。

# **1 数据集介绍** 我们采用的数据是电力变压器数据集 (ETDataset)，数据集提供了两年的数据，每个数据点每分钟记录一次（用m标记），它们分别来自中国同一个省的两个不同地区，分别名为ETT-small-m1和ETT-small-m2。每个数据集包含2年 \* 365天 \* 24小时 \* 4 = 70,080数据点。 此外，还提供一个小时级别粒度的数据集变体使用（用h标记），即ETT-small-h1和ETT-small-h2。 每个数据点均包含8维特征，包括数据点的记录日期、预测值“油温”以及6个不同类型的外部负载值，这里使用数据集ETT-small-h1。

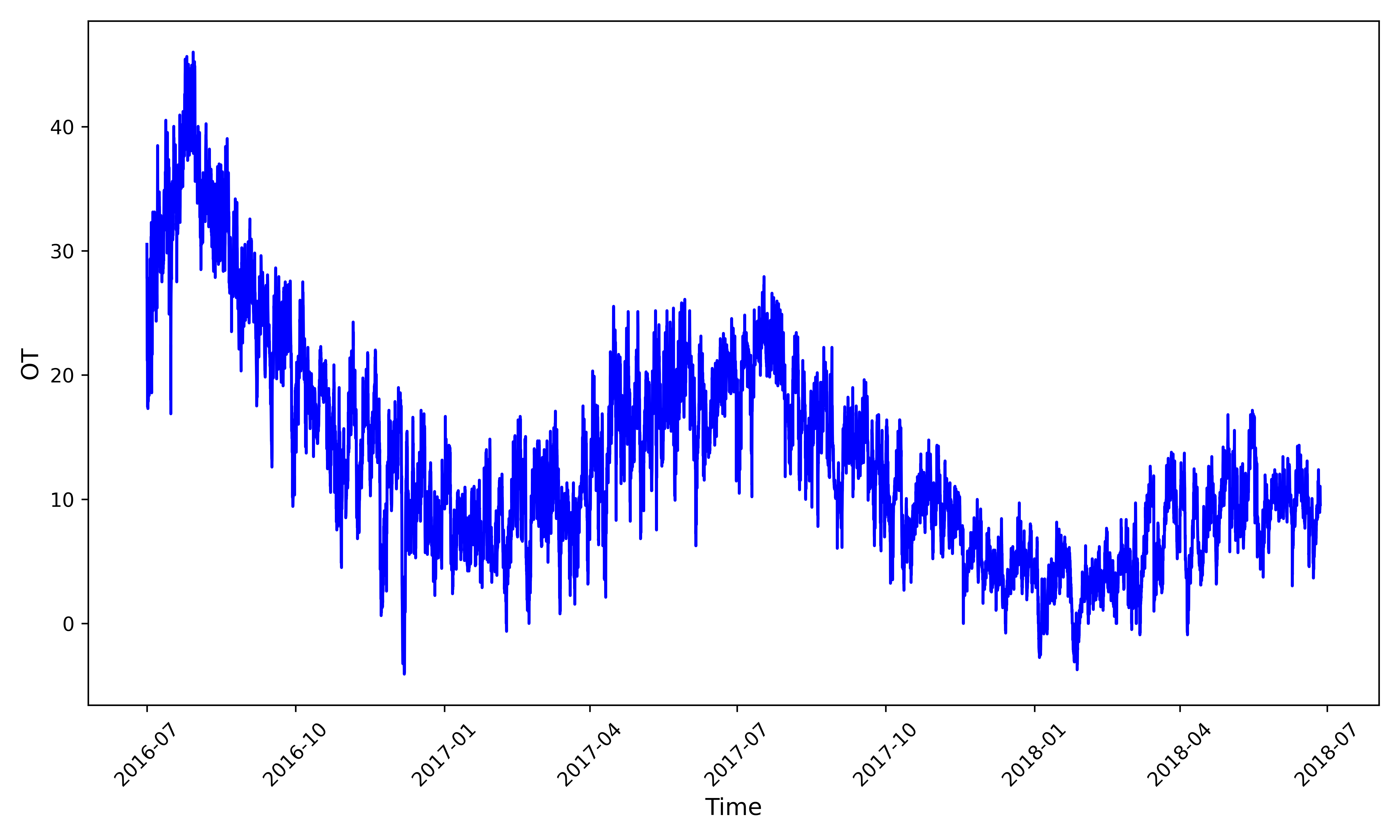


图 1’油温’(OT)两年内的变化总览

Figure 1 Overview of the changes in‘Oil Temperature’(OT)over two years

# 数据集中包含短周期模式，长周期模式，长期趋势和大量不规则模式。在图1给出了数据OT的总览，图中数据显示出了明显的季节趋势。如图2(如下)中绘制了ETT-small-h1数据集中所有变量的自相关图，其中最上面的蓝色曲线是目标变量“油温”，它保持了一些短期的局部连续性，而其他的变量（各类负载）则显示出了短期的日模式（每24小时）和长期的周模式（每7天）。

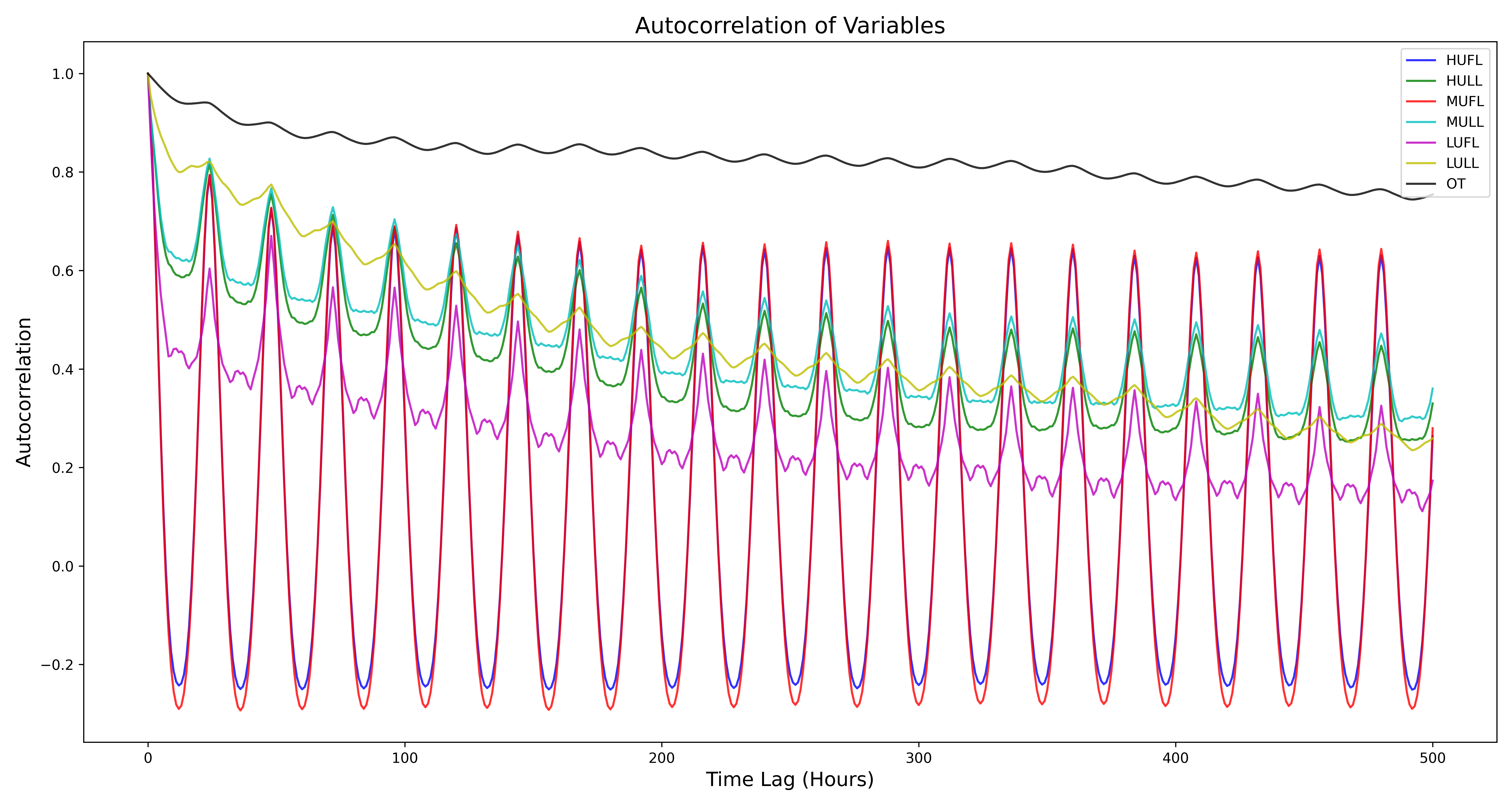


图 2各变量的自回归图

Figure 2 Autoregression plots of each variable

数据集是使用.csv形式进行存储的，其中第一行（8列）是数据头，包括了 "HUFL", "HULL", "MUFL", "MULL", "LUFL", "LULL" 和 "OT"，每一列的详细意义如表1所示。

表 1数据中各列的含义

Table 1 Meaning of each column in the data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| date | HUFL | HULL | MUFL | MULL | LUFL | LULL | OT |
| 记录日期 | **High UseFul Load** | **High UseLess Load** | **Middle UseFul Load** | **Middle UseLess Load** | **Low UseFul Load** | **Low UseLess Load** | **O**il **T**emperature (target) |

数据集是使用.csv形式进行存储的，在图3(如下)中给出一个数据的样例。

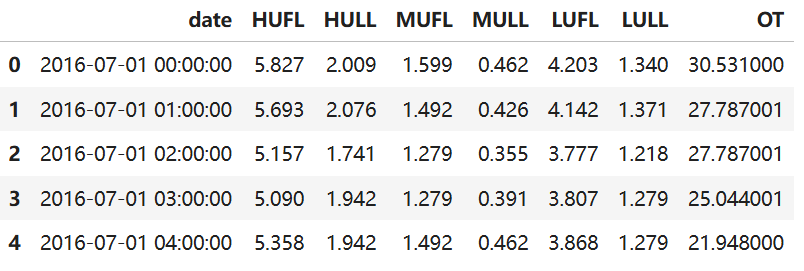


图 3ETTh1数据样例

Figure 3 ETTh1 data sample

2 模型的建立与求解  
 多元时间序列是我们日常生活中常见的现象，包括股票市场中不同股票的价格序列,以及不同城市的温度和降雨量等。顾名思义，多元时间序列的首要特征是按照时间顺序采样的具有时序性的序列数据。这类数据广泛应用于金融市场、气候预测等领域。通过对系统中多个时间依赖变量的研究，可以分析其内在的规律性和不规律性，以预测未来的趋势。

本文研究的油温同样是以多元时间序列为研究对象，涉及与油温相关的高有效负载、高无用负载等数据为预测因子，油温列为预测目标。

本文我们采用 RNN（循环神经网络）与 LSTM 预测算法对各类台风数据时间序列构建预测模型。重要实验参数设置如下表 10：

**表 2重要实验参数设置**

Table 2 Important experimental parameter settings

|  |  |
| --- | --- |
| 参数 | 参数值 |
| Batchsize | 64 |
| Dropout | 0.3 |
| 激活函数 | Tanh |
| 早停参数 | 10 |
| Epoch | 50 |
| 优化器 | Adam |
| 损失函数 | MSE |

在本文中，batch\_size（批大小）指在每次迭代中用于训练的样本数。通过采用mini-batch 的方式，可以在模型训练时有效提高训练速度和优化效率。mini-batch 方法的优势在于其能够在每次迭代中利用部分样本进行参数更新，相较于全量数据训练更为高效，且比单样本训练更稳定。通过使用RandomizedSearchCV进行超参数优化，本文选取的batch\_size大小为 64。

在本文中，Epoch（训练轮次）指的是所有数据在神经网络上完成一次完整的前向和反向传播的过程。通常，模型的训练需要多轮迭代，以便通过不断更新参数充分学习数据中的潜在信息，从而获得最优模型。训练轮次不能过少，以避免模型欠拟合；但训练过多则可能导致模型过拟合，即过度拟合训练数据，难以泛化到新数据。本文初始选择的Epoch数为100轮。

早停（Early Stopping）参数的设置旨在解决 Epoch 数量的手动设置问题。我们知道，Epoch 轮次过少可能导致模型欠拟合，过多则容易过拟合。早停法的核心目标是防止过拟合，因此可以看作是一种正则化方法[6]。其原理是，在每个Epoch结束后，用测试集评估模型性能；如果性能优于之前最佳模型，就将当前模型保存为最优模型。本文采用了max\_patience参数来控制早停。具体来说，当某一轮的损失未比前一轮更低时，继续观察后续的 patience个Epoch；如果在此期间损失没有进一步下降，则停止训练。本文中patience设为10，即如果10个Epoch 内没有性能提升，则训练将提前停止。激活函数的作用在于帮助模型学习和理解复杂的非线性信息。

在本文中，第一个全连接层后采用了tanh激活函数，以有效地提取非线性特征。tanh激活函数的输出范围为-1到1，相比于ReLU，tanh函数能够使模型在训练过程中保持更平滑的梯度，有助于避免梯度消失问题，并且可以在一定程度上减少过拟合现象。通过将激活函数选择为tanh，本文旨在更好地捕捉台风路径预测中的复杂非线性关系，提升模型的性能和稳定性。

本文的学习率优化选用了Adam算法。选择Adam的主要原因在于传统的随机梯度下降方法在训练过程中可能遇到局部最优或鞍点，从而影响模型的收敛效果。虽然目前SGD已经引入了一阶动量以帮助跳过这些特殊情况，但它无法根据每个参数的重要性来实现差异化的更新步长。相比之下，Adam算法结合了SGD的一阶动量和RMSProp 的二阶动量，使其在学习过程中具有更好的自适应性，对不同参数进行适当的调整。基于MSE这些优点，本文选用 Adam 优化算法。

2.1 **性能评价**

本文对RNN及LSTM模型的性能 采用均方误差函数（MeanSquareError，MSE）和平均绝对误差（MeanAbsoluteError，MAE）表示预测油温和真实油温数据之间的差异。

MSE的计算公式为：

(1)

MAE的计算公式为：

(2)

2.2 RNN预测模型构建与求解

循环神经网络（Recurrent Neural Network，RNN）是一种特殊的神经网络模型，具有与传统神经网络不同的特性。传统神经网络通常由输入层、隐藏层和输出层构成，且同一层的神经元之间不进行相互通信，当前时刻的输出与上一时刻的隐藏状态没有直接关系。下图(图4)展示了传统神经网络的结构[2]：

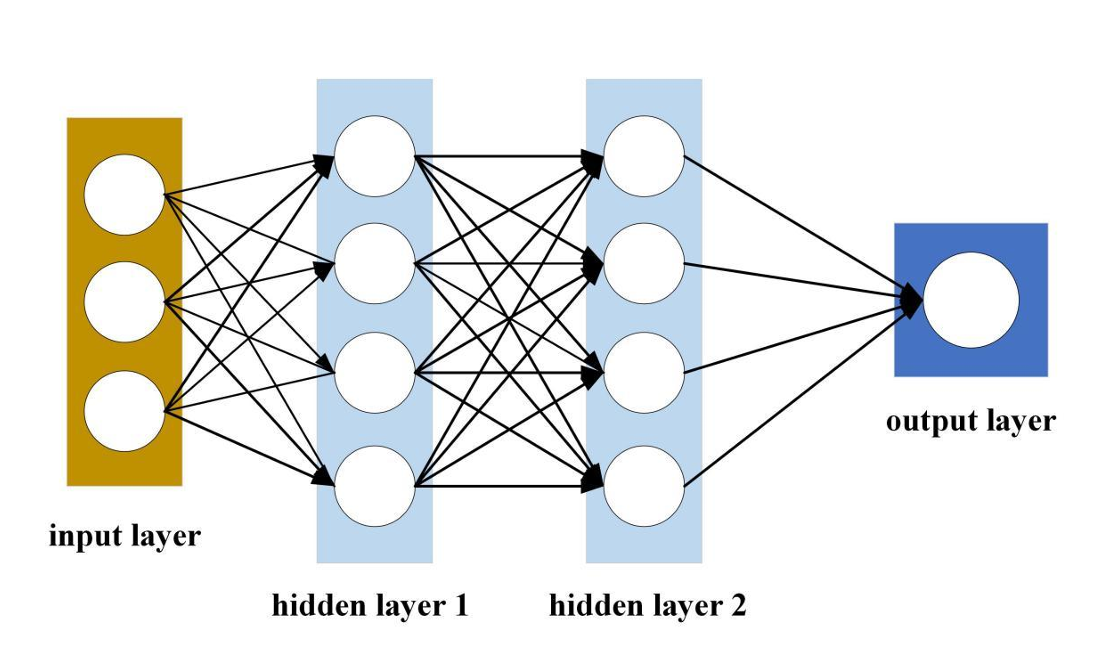


图 4RNN神经网络模型

Figure 4 RNN neural network model

ETTH1预测OT油温数据属于多元时间序列数据，具有明显的自相关性。相邻时间点的油温信息和相关的特征数据，对于预测未来的OT油温具有不同程度的相关性。RNN 由于其独特的结构，能够使所有隐藏层的神经元之间进行相互通信，使得上一次的结果以数据形式存储在隐藏层中。当下一时刻传递消息时，前一时刻的隐藏结果也会影响当前输出。通过持续这一过程，最终输出将包含前面所有的信息。这种信息传递方式体现了时间序列数据的时序特性。通用的 RNN 模型如图5所示，其中表示在时刻的输出层输出；表示在时刻的隐藏状态值；表示在 时刻的输入值；表示偏置项; 表示从隐藏层到输出层的权重矩阵；表示从隐藏层到隐藏层的权重矩阵; 表示从输入层到隐藏层的权重矩阵。

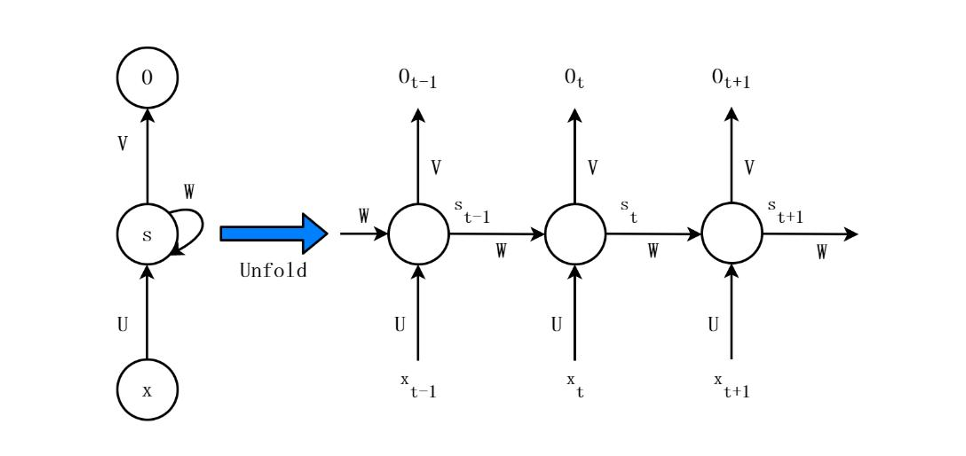


图 5RNN网络结构

Figure 5 RNN network structure

公式表示如下：

(3)

(4)

(5)

从上图的RNN结构可以看出，RNN具有两个显著特点。一方面，由于其串联结构，体现了“前因后果”的关系，后续结果的生成依赖于前面的信息，因此网络的输出与历史信息密切相关。另一方面，所有特征共享同一套参数，这不仅确保了信息传递的公平性，还大大减少了训练参数的数量。然而，普通RNN由于其串联结构，在处理长序列问题时容易遇到瓶颈，因此衍生出了一系列RNN的变形优化。

利用RNN模型对油温预测与历史实际油温对比效果如下图:

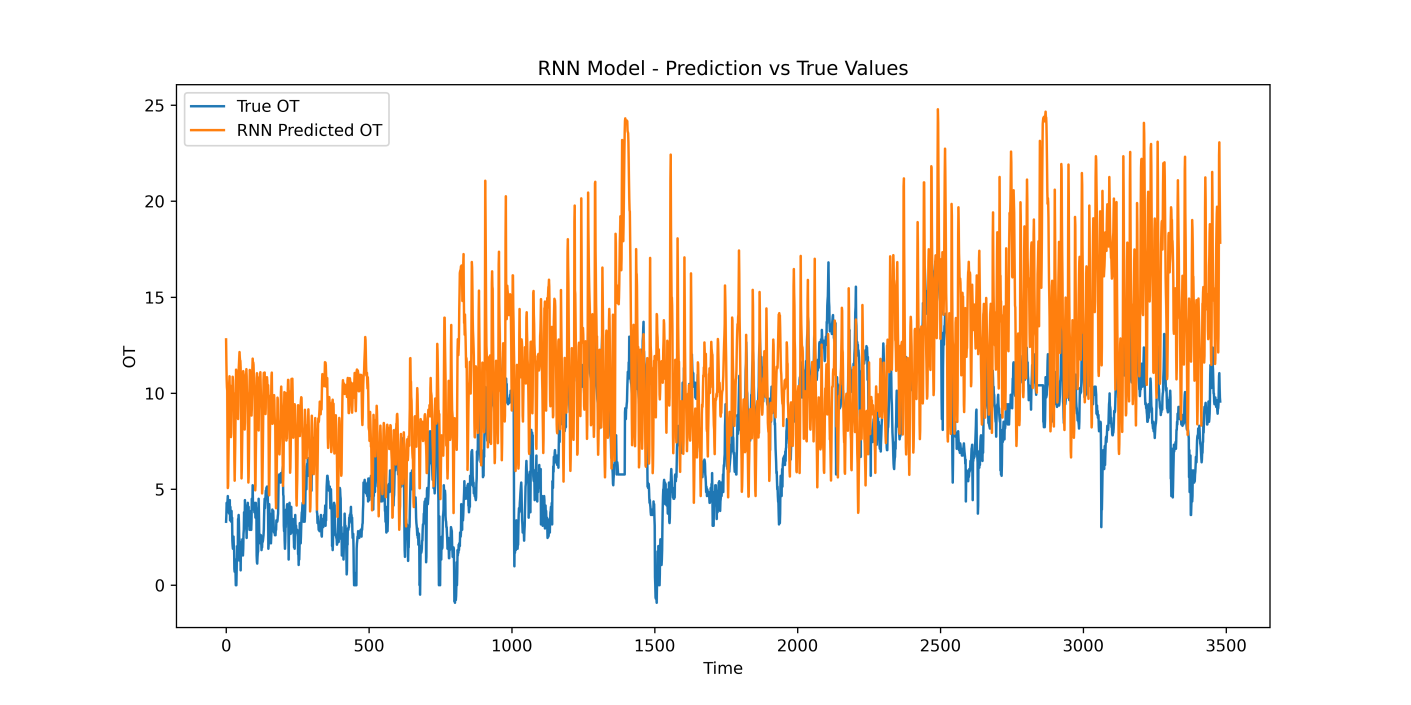


图 6RNN预测油温与实际油温对比

Figure 6 Comparison between RNN predicted oil temperature and actual oil temperature

在经度训练过程中，模型的损失值从初始的0.2051逐渐减少至0.0084，并逐步趋于稳定。在训练过程中，最大 loss下降幅度出现在第一轮训练，下降了0.1557。此次训练总共进行了31轮，在模型的loss值不再显著下降后，自动触发早停机制，停止训练，同时模型的MSE与MAE分别为30.4445与5.5177。

## 2.3 LSTM预测模型构建与求解

在预测任务中，LSTM（长短期记忆网络，Loog Short Term Memory）是一种非常常用的预测方法，尤其适用于时间序列数据。与传统的前馈神经网络不同，LSTM 是一种递归神经网络（RNN）的特殊变体，能够在较长时间内保留并利用历史信息。它通过门控机制对长期和短期信息进行筛选和保留，能有效避免梯度消失或爆炸问题，使得模型在较长时间依赖的任务上具有更好的表现。本文电力变压器数据具备明显的时间序列特征，数据记录点之间存在时间依赖性，因此使用 LSTM 模型来预测油温可以具有较高的合理性和效果预期[3]。LSTM 能够充分利用历史数据，提取其中的时序模式，从而提供对未来油温数据的有效预测。

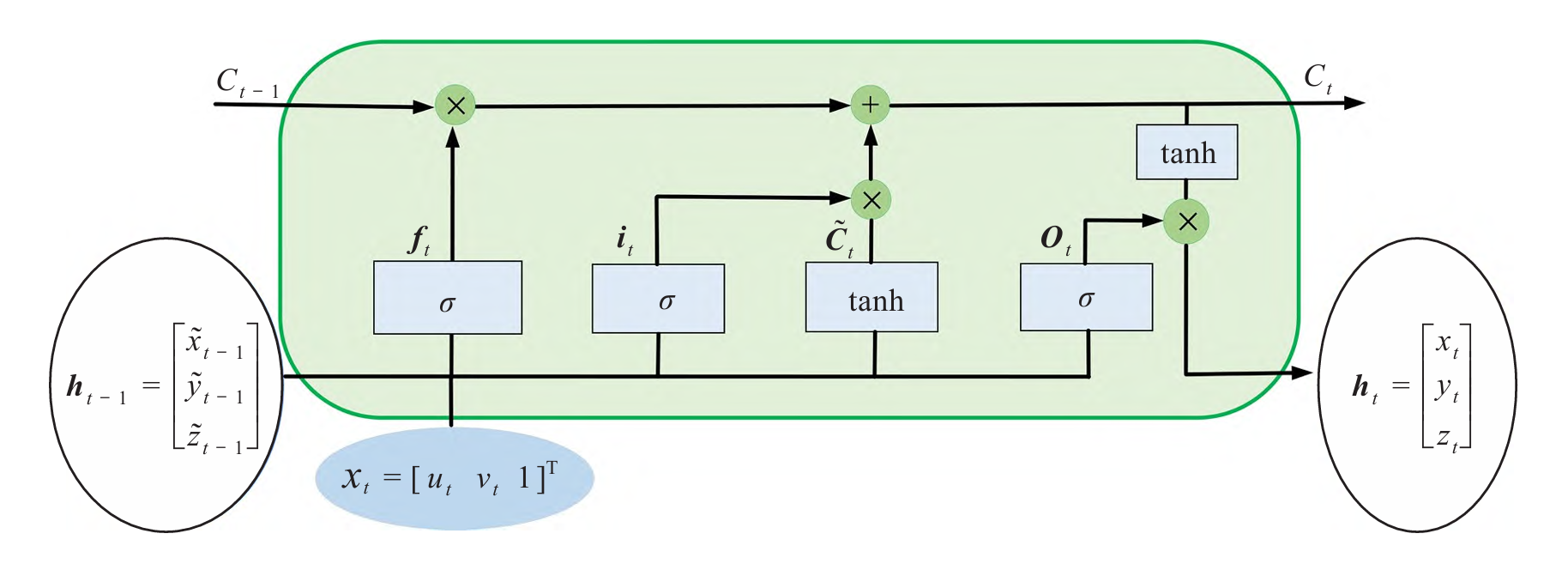


图 7 LSTM网络结构示意图

Figure 7 Schematic diagram of LSTM network structure

整个LSTM 网络模型学习过程的前向传递包 含了遗忘门、输入门和输出门。其中遗忘门用来控制时刻的记忆单元状态是否被保留，表达式为：

(6)

式中,为时刻的输入向量,为时刻的隐藏状态向量,为Sigmoid激活函数,和分别为遗忘门的权重矩阵与偏置向量。用于控制时刻的输入向量是否被更新到时刻的记忆单元更新值，分别表示为：

(7)

(8)

式中,和为输入门的权重矩阵与偏置向量，和为记忆单元更新值的权重矩阵 与偏置向量。

时刻的记忆单元状态通过遗忘门和输入门控制时刻的记忆单元状态以及更新值得到：

(9)

基于更新时刻的隐藏状态向量，表达式分别为：

(10)

(11)

式中，和为输出门的权重矩阵与偏置向量。

LSTM通过不同的“门”来控制抛弃或者增加信息，从而实现遗忘或记忆的功能，模拟神经网络。“门”是一种使信息选择性通过的结构，由一个 Sigmoid函数和一个点乘操作组成。Sigmoid函数为生物学中常用的S型函数，其值域在[0，1]区间，当“门”输出为0时代表此信息完全丢弃，为1时代表完全通过。一个LSTM单元有三个这样的门，分别是遗忘门、输入门、输出门。简单来说，LSTM中的三个门就是通过三个[0，1]之间的函数来有选择的过滤旧的信息，从而加入新的信息。

本文通过Python编程来实现LSTM对数据进行预测，将电力变压器各类特征数据作为输入，油温数据即OT列作为输出。从而利用LSTM神经网络对电力变压器数据进行训练，最终将训练得到的模型对油温进行预测。结果如图7：

|  |  |
| --- | --- |
| LSTM_loss_curve | LSTM预测结果 |
| (a)LSTM训练损失图 | (b)LSTM预测结果 |

图 8 LSTM模型训练与预测结果

Figure 8 LSTM model training and prediction results

((a)LSTM training loss graph; (b)LSTM prediction results)

在经度训练过程中，模型的损失值从初始的0.0340逐渐减少至0.0062，并逐步趋于稳定。在训练过程中，最大 loss下降幅度出现在第一轮训练，下降了0.061。此次训练总共进行了28轮，在模型的loss值不再显著下降后，自动触发早停机制，停止训练，同时模型的MSE与MAE分别为25.7840与5.0778。

# 4 结 论 本文利用神经网络模型RNN与长短时记忆网络LSTM对可靠数据集ETDataset中的ETT-small-h1电力变压器数据进行了油温(OT)预测，发现ETTh1中各项数据包含短周期模式，长周期模式，长期趋势和大量不规则模式，油温”保持了一些短期的局部连续性，而其他变量（各类负载）则显示出了短期的日模式（每24小时）和长期的周模式，对于模型预测，LSTM模型的预测结果较好于RNN模型，模型展现出了较好的预测能力，这与我们的预期相符。在后期工作中我们可以采取其他对长时间序列有较好预测能力的模型如Informer、Dlinear、LSTF等进行赋权融合提高整体的预测能力。

#### 参考文献(References)

1. Zhou HY. 2021.The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference. IEEE：11106-11115
2. Robert L.2020.Python Data Mining Introduction and Practice(Python数据挖掘入门与实践(第2版).[M].北京:人民邮电出版社,2020.)
3. Zhang Q Q.Research.2024. Chongqing University of Technology.Research on time series prediction based on causal convolution and attention mechanism(张青青.基于因果卷积与注意力机制的时间序列预测研究[D].重庆理工大学,2024.)DOI:10.27753/d.cnki.gcqgx.2024.001340.

#### 代码电子附录（Code）

1. **import** pandas as pd
2. **import** numpy as np
3. **from** sklearn.preprocessing **import** MinMaxScaler
4. **from** sklearn.model\_selection **import** train\_test\_split
5. **import** matplotlib.pyplot as plt
6. **from** tensorflow.keras.models **import** Sequential
7. **from** tensorflow.keras.layers **import** LSTM, Dense
8. **from** statsmodels.graphics.tsaplots **import** plot\_acf
9. **from** statsmodels.tsa.stattools **import** acf
11. # 假设数据存储在一个 CSV 文件中
12. df = pd.read\_csv(r"C:\Users\z\Desktop\数据挖掘\ETTh1油温预测\ETTh1.csv")
13. df.head(5)
14. # 转换日期列为 datetime 格式
15. df['date'] = pd.to\_datetime(df['date'])
17. # 设置日期为索引
18. df.set\_index('date', inplace=True)
19. df
20. # 绘制 OT 列随时间变化的图
21. plt.figure(figsize=(10, 6))
22. plt.plot(df.index, df['OT'], label='OT', color='blue')
23. plt.xlabel('Time', fontsize=12)
24. plt.ylabel('OT', fontsize=12)
25. plt.xticks(rotation=45)
26. #plt.legend('OT',loc='upper right',fontsize=20)
27. plt.tight\_layout()
28. # 保存 OT 图
29. plt.savefig("OT\_时间变化图.png",dpi=600)
30. # 设置颜色列表
31. colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'orange', 'purple', 'brown']
33. # 创建一个更紧凑的子图布局
34. plt.figure(figsize=(15, 8))
36. # 在一个图中绘制所有变量的自回归
37. **for** i, col **in** enumerate(df.columns):
38. acf\_values = acf(df[col], nlags=500, fft=True)
39. plt.plot(acf\_values, label=col, color=colors[i % len(colors)], alpha=0.8)
41. # 设置标题和图例
42. plt.title('Autocorrelation of Variables', fontsize=16)
43. plt.xlabel('Time Lag (Hours)', fontsize=14)
44. plt.ylabel('Autocorrelation', fontsize=14)
45. plt.legend(loc='upper right', fontsize=10)
47. # 调整布局并保存图片
48. plt.tight\_layout()
49. plt.savefig("各变量自回归图\_无竖线.png", dpi=600)
50. plt.show()
51. **import** numpy as np
52. **import** matplotlib.pyplot as plt
53. **from** tensorflow.keras.models **import** Sequential
54. **from** tensorflow.keras.layers **import** Input, SimpleRNN, Dropout, Dense
55. **from** tensorflow.keras.callbacks **import** EarlyStopping
56. **from** sklearn.metrics **import** mean\_squared\_error, mean\_absolute\_error, r2\_score
58. # 假设 scaler\_y 已经定义，确保你在代码中有合适的归一化部分
59. # scaler\_y = ...
61. # 定义 RNN 模型
62. **def** create\_optimized\_rnn\_model(input\_shape):
63. model = Sequential()
64. model.add(Input(shape=input\_shape))
65. model.add(SimpleRNN(units=64, activation='tanh', return\_sequences=True))
66. model.add(Dropout(0.3))  # 添加 Dropout 层防止过拟合
67. model.add(SimpleRNN(units=64, activation='tanh'))  # 第二个 RNN 层
68. model.add(Dropout(0.3))  # 添加 Dropout 层
69. model.add(Dense(1))  # 输出预测值
70. model.compile(optimizer='adam', loss='mse')
71. **return** model
73. # 创建 RNN 模型
74. optimized\_rnn\_model = create\_optimized\_rnn\_model(input\_shape=(X\_train.shape[1], X\_train.shape[2]))
75. optimized\_rnn\_model.summary()
77. # 早停机制
78. early\_stopping = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)
80. # 训练 RNN 模型
81. optimized\_rnn\_model.fit(X\_train, y\_train, epochs=50, batch\_size=64, validation\_data=(X\_test, y\_test), callbacks=[early\_stopping])
83. # 使用模型进行预测
84. y\_pred\_optimized\_rnn = optimized\_rnn\_model.predict(X\_test)
86. # 反归一化预测值
87. y\_pred\_optimized\_rnn\_rescaled = scaler\_y.inverse\_transform(y\_pred\_optimized\_rnn)
89. # 反归一化真实值
90. y\_test\_rescaled = scaler\_y.inverse\_transform(y\_test.reshape(-1, 1))
92. # 打印预测的形状
93. **print**(f'Predicted shape: {y\_pred\_optimized\_rnn\_rescaled.shape}')
94. **print**(f'Real shape: {y\_test\_rescaled.shape}')
96. # 绘制并保存 RNN 预测结果
97. plt.figure(figsize=(12, 6))
99. # 绘制真实值
100. plt.plot(y\_test\_rescaled, label='True OT')
102. # 绘制优化后的 RNN 预测值
103. plt.plot(y\_pred\_optimized\_rnn\_rescaled, label='RNN Predicted OT')
105. plt.legend()
106. plt.title('RNN Model - Prediction vs True Values')
107. plt.xlabel('Time')
108. plt.ylabel('OT')
110. # 保存图像
111. plt.savefig('RNN预测结果', dpi=600)
112. plt.show()
114. # 计算均方误差 MSE 和均方根误差 RMSE
115. mse = mean\_squared\_error(y\_test\_rescaled, y\_pred\_optimized\_rnn\_rescaled)
116. rmse = np.sqrt(mse)
118. # 计算平均绝对误差 MAE
119. mae = mean\_absolute\_error(y\_test\_rescaled, y\_pred\_optimized\_rnn\_rescaled)
121. # 打印评估指标
122. **print**(f'Mean Squared Error (MSE): {mse}')
123. **print**(f'Root Mean Squared Error (RMSE): {rmse}')
124. **print**(f'Mean Absolute Error (MAE): {mae}')
126. # 创建时间步数据集（使用过去 24 小时数据来预测未来 24 小时）
127. **def** create\_dataset(X, y, time\_step=24):
128. X\_data, y\_data = [], []
129. **for** i **in** range(len(X) - time\_step):
130. X\_data.append(X[i:i+time\_step])
131. y\_data.append(y[i+time\_step])
132. **return** np.array(X\_data), np.array(y\_data)
133. # 创建训练数据集
134. X\_data, y\_data = create\_dataset(X\_scaled, y\_scaled, time\_step=24)
135. # 将数据分为训练集和测试集
136. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_data, y\_data, test\_size=0.2, shuffle=False)
137. # 打印数据的形状
138. **print**(f'X\_train shape: {X\_train.shape}')
139. **print**(f'X\_test shape: {X\_test.shape}')
140. # 定义LSTM模型
141. **def** create\_optimized\_lstm\_model(input\_shape):
142. # 使用 Input 层来定义输入形状
143. inputs = Input(shape=input\_shape)
145. # LSTM 层
146. x = LSTM(units=128, activation='tanh', return\_sequences=True)(inputs)
147. x = Dropout(0.2)(x)  # 添加 Dropout 层防止过拟合
148. x = LSTM(units=64, activation='tanh')(x)  # 第二个 LSTM 层
149. x = Dropout(0.2)(x)  # 添加 Dropout 层
150. output = Dense(1)(x)  # 输出预测值
152. # 创建模型
153. model = Model(inputs=inputs, outputs=output)
155. model.compile(optimizer='adam', loss='mse')
156. **return** model
158. # 创建 LSTM 模型
159. optimized\_lstm\_model = create\_optimized\_lstm\_model(input\_shape=(X\_train.shape[1], X\_train.shape[2]))
160. # 显示模型结构
161. optimized\_lstm\_model.summary()
162. # 可视化并保存模型结构图
163. plot\_model(optimized\_lstm\_model, to\_file='lstm\_model\_structure.png', show\_shapes=True, show\_layer\_names=True)
164. # 早停机制
165. early\_stopping = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)
166. # 训练 LSTM 模型
167. history = optimized\_lstm\_model.fit(X\_train, y\_train, epochs=100, batch\_size=64, validation\_data=(X\_test, y\_test), callbacks=[early\_stopping])
168. # 使用模型进行预测
169. y\_pred\_optimized\_lstm = optimized\_lstm\_model.predict(X\_test)
170. # 反归一化预测值
171. y\_pred\_optimized\_lstm\_rescaled = scaler\_y.inverse\_transform(y\_pred\_optimized\_lstm)
172. # 反归一化真实值
173. y\_test\_rescaled = scaler\_y.inverse\_transform(y\_test.reshape(-1, 1))
174. # 打印预测的形状
175. **print**(f'Predicted shape: {y\_pred\_optimized\_lstm\_rescaled.shape}')
176. **print**(f'Real shape: {y\_test\_rescaled.shape}')
177. # 绘制并保存 LSTM 预测结果
178. plt.figure(figsize=(12, 6))
179. # 绘制真实值
180. plt.plot(y\_test\_rescaled, label='True OT')
181. # 绘制优化后的 LSTM 预测值
182. plt.plot(y\_pred\_optimized\_lstm\_rescaled, label='LSTM Predicted OT')
183. plt.legend()
184. plt.title('LSTM Model - Prediction vs True Values')
185. plt.xlabel('Time')
186. plt.ylabel('OT')
187. # 保存图像
188. plt.savefig('LSTM预测结果', dpi=600)
189. plt.show()
191. # 计算均方误差 MSE 和均方根误差 RMSE
192. mse = mean\_squared\_error(y\_test\_rescaled, y\_pred\_optimized\_lstm\_rescaled)
193. rmse = np.sqrt(mse)
194. # 计算平均绝对误差 MAE
195. mae = mean\_absolute\_error(y\_test\_rescaled, y\_pred\_optimized\_lstm\_rescaled)
197. # 打印评估指标
198. **print**(f'Mean Squared Error (MSE): {mse}')
199. **print**(f'Root Mean Squared Error (RMSE): {rmse}')
200. **print**(f'Mean Absolute Error (MAE): {mae}')
202. # 获取训练和验证集的损失值
203. train\_loss = history.history['loss']
204. val\_loss = history.history['val\_loss']
206. # 绘制损失函数曲线
207. plt.figure(figsize=(12, 6))
208. plt.plot(train\_loss, label='Train Loss')
209. plt.plot(val\_loss, label='Validation Loss')
211. plt.title('LSTM Model - Training and Validation Loss')
212. plt.xlabel('Epochs')
213. plt.ylabel('Loss')
214. plt.legend()
216. # 保存损失函数曲线
217. plt.savefig('LSTM\_loss\_curve', dpi=600)
218. plt.show()