兰州大学

本科生毕业论文

论文题目（中文） 基于 ICEEMDAN 多特征分解和 Prophet-GRU-NN 组合模型多步预测短期风速

论文题目（英文） Multistep-Ahead Forecast Short-Term Wind Speed through ICEEMDAN Decomposed Multi-features and Prophet-GRU-NN Hybrid Model

学生姓名 蒋嵩林

指导教师 任超

学　　院 信息科学与工程学院

专　　业 计算机科学与技术（基础理论班）

年　　级 2018 级

兰州大学教务处

诚信责任书

本人郑重声明：本人所呈交的毕业论文（设计），是在导师的指导下独立进行研究所取得的成果。毕业论文（设计）中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，不包含任何其他个人或集体已经发表或在网上发表的论文。

本声明的法律责任由本人承担。

论文作者签名：蒋嵩林　　日　　期：_____

关于毕业论文（设计）使用授权的声明

本人在导师指导下所完成的论文及相关的职务作品，知识产权归属兰州大学。本人完全了解兰州大学有关保存、使用毕业论文（设计）的规定，同意学校保存或向国家有关部门或机构送交论文的纸质版和电子版，允许论文被查阅和借阅；本人授权兰州大学可以将本毕业论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业论文（设计）。本人离校后发表、使用毕业论文（设计）或与该毕业论文（设计）直接相关的学术论文或成果时，第一署名单位仍然为兰州大学。

本毕业论文（设计）研究内容：

☑可以公开

☐ 不宜公开，已在学位办公室办理保密申请，解密后适用本授权书。

（请在以上选项内选择其中一项打“√”）

论文作者签名：蒋嵩林　　导师签名：_____

日　　期：_____　　日　　期：_____

毕业论文（设计）成绩表

导师评语 **建议成绩**_____ **指导教师（签字）**_____
答辩小组意见 **答辩委员会负责人（签字）**_____
成绩_____ **学院（盖章）**_____ 年 月 日

基于 ICEEMDAN 多特征分解和 Prophet-GRU-NN 组合模型多步预测短期风速

中文摘要

气候变化和环保问题关乎人类未来命运。在石油危机以及全球变暖问题愈发严重的现在，可再生清洁能源相关研究成为了学界的关注热点。风能，作为触手可得的一种能源，正成为新能源的主力军之一。准确预测风速，从而保证电网的稳定性，具有十分重大的意义。

风速预测是一个时间序列回归问题，由于风速的波动具有随机性，其影响因素也十分复杂，呈现出非平稳且非线性的特征，因而对短期风速的预测难度较大。本文基于前人研究成果，选用 ICEEMDAN 分解技术，并使用当前最新循环神经网络结构 GRU，结合 Facebook Prophet 框架，形成 ICEEMDAN-Prophet-GRU 模型。同时结合数值预报方法提供的灵感，将其他天气要素特征也输入到模型中，最后使用神经网络（NN）整合校正，提高模型的泛化能力。

最终结果表明，使用本文提出的多特征 ICEEMDAN-Prophet-GRU-NN 模型，预测短期风速的精度会明显提升，因而具有优越性。

关键词：风速影响因素，短期风速预测，组合模型，改进自适应噪声完备集合经验模态分解，Prophet，门控循环单元神经网络，深度学习，数据科学

MULTISTEP-AHEAD FORECAST SHORT-TERM WIND SPEED THROUGH ICEEMDAN DECOMPOSED MULTI-FEATURES AND PROPHET-GRU-NN HYBRID MODEL

Abstract

Climate change and environmental issues matter the destiny of humankind. With the oil crisis and global warming worsening, renewable energy-related research has become a hot spot attracting great academic attention. Wind is becoming mainstream renewable energy as a readily available energy source. As a result, it is essential to predict the wind speed accurately to ensure power system stability.

Wind speed forecast is a time series regression problem. Due to the randomness of wind speed fluctuations, influencing factors for wind speed are very complex. The wind speed has non-stationary and nonlinear characteristics, so it is difficult to predict short-term wind speed. Based on the previous research results, this dissertation selects the ICEEMDAN decomposition technology, and uses the latest recurrent neural network model GRU, combined with the Facebook Prophet framework, to form the ICEEMDAN-Prophet-GRU model. At the same time, combined with the inspiration provided by the numerical weather forecast method, other common meteorological elements are also inputted into the model. Finally the neural network (NN) is used to integrate and correct, so that the generalization ability of the model can be improved.

The final results show that using the multi-feature ICEEMDAN-Prophet-GRU-NN model proposed in this paper, we can significantly improve the accuracy of short-term wind speed prediction, so it has supremacy.

Key Words: wind speed influencing factors; short-term wind speed forecast; hybrid model; ICEEMDAN; Prophet; GRU; deep learning; data science.

目　录

中文摘要 I
英文摘要 II
第一章　绪　　论 1
1.1　选题研究背景及意义 1
1.1.1　风力发电的前景 1
1.1.2　短期风速预测的意义 1
1.2　选题已有研究成果综述 2
1.2.1　数值预报方法 2
1.2.2　统计与传统机器学习方法 2
1.2.3　深度学习方法 3
1.3　论文研究内容与组织结构 4
第二章　数据来源及特征选取 6
2.1　数据来源 6
2.2　特征选取与数据集的制作 6
2.3　特征数据分析 7
2.3.1　描述性统计分析 7
2.3.2　相关性分析和显著性检验 8
第三章　使用技术的介绍和比较 9
3.1　基于 EMD 的信号分解技术 9
3.1.1　EMD 9
3.1.2　EEMD 10
3.1.3　CEEMDAN 10
3.1.4　ICEEMDAN 11
3.2　基于 Prophet 框架的时间序列预测方法 11

3.3 基于 PCA 的降维方法 11
3.3.1 主成分分析 11
3.3.2 核主成分分析 12
3.4 基于 RNN 的深度学习神经网络模型 12
3.4.1 经典循环神经网络原理 12
3.4.2 GRU 原理 13
3.4.3 神经网络模型激活函数 14
3.4.4 神经网络模型优化器 16
3.4.5 神经网络模型损失函数 17
第四章 模型的建立和评估 20
4.1 六种特征数据的 ICEEMDAN 分解 20
4.2 搜寻 Prophet 模型最优超参数 26
4.3 基于风速历史时间序列数据的 ICEEDMAN-GRU 模型 28
4.4 基于风速历史时间序列数据的 ICEEDMAN-Prophet-GRU 模型 30
4.5 多特征 ICEEDMAN-Prophet-GRU-NN 模型 32
4.6 模型评价指标公式 34
4.7 模型的对比评估 34
第五章 总结与展望 37
参考文献 38
附 录 40
A.1 数据集制作代码 40
A.2 表 2.2 中的数据源制作代码 41
A.3 表 2.3 中的数据源制作代码 41
A.4 图 4.1-图 4.6 制作代码以及 ICEEMDAN 分解数据的存储 41
A.5 基于风速历史时间序列数据的 Prophet 模型的参数交叉验证调优 ... 43
A.6 模型评价指标实现代码 44
A.7 基于风速历史时间序列数据的 Prophet 模型实现代码和相关指标计算、可视化对比图形的制作 44

A.8 基于风速历史时间序列数据的 ICEEDMAN-Prophet 模型实现代码和相关指标计算、可视化对比图形的制作 45
A.9 基于风速历史时间序列数据的 ICEEDMAN-GRU 模型实现代码和相关指标计算、可视化对比图形的制作 45
A.10 基于风速历史时间序列数据的 ICEEDMAN-Prophet-GRU 模型实现代码和相关指标计算、可视化对比图形的制作..... 47
A.11 多特征 ICEEDMAN-Prophet-GRU-NN 模型实现代码实现代码和相关指标计算、可视化对比图形的制作 50
A.12 文中使用的函数图像绘制代码 54
致　　谢 57

第一章 绪 论

1.1 选题研究背景及意义

1.1.1 风力发电的前景

随着工业发展的需要，人类社会对能源的依赖度也在逐步上升。第二次能源革命以来，传统化石燃料的使用导致了一系列的问题。化石燃料在开采过程中也产生了一系列的环境问题。以煤炭为例，在开采时，会破坏地表原本的性状，引发滑坡，塌陷等一系列问题。开采煤炭所产生的废渣也难以处理，产生的污水会对水土环境造成污染。同时，燃烧化石燃料所产生的二氧化硫 (SO_2) 和氮氧化物会形成酸雨，粉尘使空气能见度降低，一氧化碳 (CO) 和芳香烃化合物会污染空气；二氧化碳 (CO_2) 在大气层中所占比例的提高，也导致了温室效应与气候变化，从而使得一些极端天气灾害的发生频率有了显著的升高。然而目前，在世界能源消费占比中，煤炭和石油的比重仍较多。我国的能源生产和消费构成中，煤炭在 2019 年仍在一次能源中占比 57.7%，占据着主要地位。[1]

当前传统化石燃料，由于其储量的有限性，也正逐步面临枯竭的风险。“绿水青山就是金山银山”。可再生能源的发展，不仅将会确保绿水青山的存在，而且同时也必然即将成为金山银山的重要组成部分。过去三十年以来，风能一直保持着最快增长的可再生能源的记录，是目前全球第二大的可再生能源。风能本质上属于一种气象能源，其为太阳照射环境下地球表面受热不均，在水平气压梯度力的影响下，由于温差造成大气对流所产生的一种能量来源。李仲蔚的研究 [2] 估算结果表明，全球可开发利用的风能资源达到了二百万兆瓦 ($2 \times 10^7 MW$)，是目前全球第一大的可再生能源水能的 10 倍。风能因其分布广泛，弥补了水能发电需要的苛刻条件以及对生态自然环境可能造成不良影响的缺陷。如将预估的可利用风能的 1% 加以利用，即可以满足全球能源的发电使用需求，因而发展潜力极其巨大，十分有望在将来替代水能成为全球第一大的可再生能源。

1.1.2 短期风速预测的意义

短期风速的主要影响因素分为气象因素和地形因素。其中，气象因素主要包括温度、气压、湿度等，而地形因素包括地貌、地表障碍等。

对于短期近地面风速的预测能够很好的帮助并促进风能的应用。首先，风力发电厂管理人员可以通过预测结果优化电力分配，提高发电量；其次，风力发电厂还可以准确地在风速过大之前对风力发电机指示停机，预防风力发电机的过载损毁，避免或者减少相应的损失；最后，还可以有效安排好风力发电的电网并网问题，减少因为风速的剧烈波动对电网电业的稳定性影响，降低运营成本，增加风电场的效益。

对于目前的风速预测研究而言，相关问题主流使用的模型包括传统时间序列模型以及深度学习模型。他们各自都存在一定的优点和缺陷，单一模型并不能很好的进行预测 [3]。同时，由于风速数据中往往包含着较大的噪声问题，如何有效地对其数据集进行降噪处理也是目前研究的方向。由于单一的预测方法无法做到很好的风速预测，如果能通过组合模型，对其进行精确地预测，将会对风力发电的应用起到巨大的促进作用。

1.2 选题已有研究成果综述

1.2.1 数值预报方法

数值预报方法使用大气压、气温、相对湿度、风速等数值型数据来作为初始状态，以流体力学方程和热力学方程为依据来实现整个大气状态模型的构建，通过超级计算机进行数值计算求解，得到未来天气的预报结果。[4]

目前数值预报方法已经很成熟，准确性良好，已经广泛地应用到了天气预报中。但是，由于数值方法需要占用大量的计算资源，难以将时间和空间分辨率都做得很高，且短期风速中蕴含的噪声较大，因而单纯的数值预报一般不适用于短期风速预报。朱智慧等人 [5] 将基于 T639 全球中期数值预报产品得到的上海南汇站 24 小时风速预报结果利用逐步回归分析、结合 Kalman 滤波得出的 MOS 方程进行订正，结果表明其订正后的精度可以考虑投入实际使用。夏晓玲等人 [6] 利用数值预报的 ECWMF，GRAPES 和 JAPAN 模式对于贵州省范围内 84 个站点的风速预报进行了分析，最终运用神经网络对 3 种模式风速预报进行订正，结果表明其误差，正确率订正改善效果明显，但是相关系数的提高较小。

1.2.2 统计与传统机器学习方法

不同于数值预报方法，统计与传统机器学习方法基于风速历史数据的时间序列，通过训练机器学习模型来进行预测，并且可以对数据进行预处理，从而可以从历史数据中挖掘出风速数据的潜在规律，因而相对于数值预报方法，模型的训练和预测更加简单，无需使用超级计算机进行大量的计算。

a. 统计方法：

经典的统计方法中对于时间序列数据的处理一般采用自回归综合移动平均模型（ARIMA, AutoRegressive Integrated Moving Average）。其采用过去观测值的线性函数关系预测未来值。但是由于风速的时间序列数据普遍是非平稳的，包含了很强的季节性，因此相关研究普遍采用季节性自回归综合移动平均模型（SARIMA, Seasonal AutoregRessive Integrated Moving Average）。SARIMA 是 Box 和 Jenkins 提出的 ARIMA 模型的一个扩展，用于处理具有季节特征的时间序列数据。Sun, R. N. 等人 [7] 同时基于 ARIMA 和 SARIMA 模型提前 24 小时预测小时平均风速和风力发电量。并进行了平均相对误差的比较分析。结果表明，SARIMA 模型的预测效果明显优于 ARIMA 模型。

SARIMA 本身机理决定了 SARIMA 无法处理非线性时间序列特征，因而在呈现出非平稳且非线性特征的风速时间序列预测中表现并不是很好。Haddad 等人 [8] 基于 SARIMA 方法构建了一个太阳能和风能预测模型，对于风能预测而言最终效果不是很突出。Xl, A 等人 [9] 使用 SARIMA 模型来预测苏格兰沿海地区的每小时实测风速。结果表明尽管 SARIMA 在风速预测的性能与准确度在相互权衡之后相对于循环神经网络方法具有优越性，但是其预测的风速准确度并不是很理想。

b. 机器学习方法：

传统机器学习方法，在风速中应用最广泛的为支持向量回归机（SVR，Support Vector Regression）。支持向量回归机是支持向量机（SVM，Support Vector Machine）的一个非常重要的变体，其将 SVM 的应用领域从分类变成了回归。SVR 的目标是对时间序列输入数据的每个点通过非线性变换寻找一个最优超平面。不同于 SVM 的最优超平面需要使得两类或多类样本点与超平面之间的总偏差最大，SVR 的样本点最终只有一类，并且其最优超平面需要所有的样本点与超平面之间的总偏差最小。

传统的 SVR 能够对非线性特性的数据有较好的预测能力，然而由于风速呈现出非平稳的特征，其中所含噪声较多，直接对风速进行预测无法达到很好的效果。朱霄旬等人 [10] 使用一种基于遗传算法 (GA，Genetic Algorithm) 的多参数同步优化算法，改进了传统通过相空间重构法求出 SVR 预测模型的最优参数的方法，并大大提高了预测精度。Pan, C. 等人 [11] 首先根据回归率和确定性相结合的联合指数，对风速序列的可预测性进行了定量分析，然后利用联合指数优化的参数重建风速序列，通过嵌入维数和延迟时间得到预测模型的最佳输入集。最终利用布谷鸟优化算法（COA，Cuckoo Optimization Algorithm）优化的 SVR 模型对风速进行预测，取得了不错的效果。

1.2.3 深度学习方法

近年来，随着深度学习的发展，深度学习方法在风速预测中的应用越来越广泛。深度学习由数据驱动，具有十分强大的泛化能力，十分适合处理具有非平稳且非线性特征的风速时间序列预测。神经网络模型虽然预测效果好，但是存在训练难度大的问题，对数据集和参数的要求十分高，否则容易产生过拟合、梯度爆炸、梯度消失等一系列问题。

Shivani 等人 [12] 对传统时间序列统计模型 ARIMA 和一个深度学习模型预测风速进行了比较研究。该深度学习模型由长短期记忆网络（LSTM，Long Short-Term Memory）和循环神经网络（RNN，Recurrent Neural Network）组合而成，结果凸显了深度学习预测风速的优越性。Alencar, D. B. 等人 [13] 提出了一种基于 SARIMA 和反向传播（BP，Back Propagation）神经网络的组合模型方法，用于多步超前风速预测。并进行了仿真分析。其结果表明，对于不同的预测时段，该组合模型预测方法的预测精度优于大部分传统算法。朱丽娜的研究 [14] 得出结论：LSTM 适用于预测非平稳序列风速，Elman 神经网络，一种动态递归神经网络，次之。

由于风速数据的非平稳性，其时间序列数据中往往掺杂着较大的噪声。这些噪声如果不分离出来，将会一定程度上影响到深度学习模型对风速时间序列的预测精度和泛化能力，加大对其非线性关系的学习难度。许多常见的信号分解技术包括傅里叶变换以及小波变换，将时间序列信号从时域变换到频域，从而将高频噪声分离。常见的降噪方法包括小波分解（WD，Wavelet Decomposition）、变分模式分解（Variational Mode Decomposition）、Hilbert-Huang 变换（HHT，Hilbert-Huang Transform）以及经验模式分解（EMD，Empirical Mode Decomposition）以及基于上述四类方法演化出的一系列分支。

谢义超 [15] 提出了一种基于 CEEMDAN 分解和改进的 LSTM 模型。该模型基于粒子群优化算法（PSO，Particle Swarm Optimization）与样本熵选择 CEEMDAN 以及 LSTM 的超参数，从而让参数达到最优值。最终得出的结果表明，CEEMDAN 分解技术可以极大地提高基于循环神经网络模型的泛化能力，降低模型的训练难度。王秀的研究 [16] 基于小波变换，通过 SARIMA 和 LSTM 模型的组合，提升了对短期风速的预测精度。

1.3　论文研究内容与组织结构

随着数据科学的发展，使用开箱即用（Out of the box）的数据分析框架已经成为数据科学研究的一种新潮流。在这之中包括了流行的机器学习框架 scikit-learn，以及时间序列数据处理预测框架 Prophet。

由于注意到直接使用 Prophet 框架来进行非平稳非线性的风速预测效果并不佳，因而本文基于上述前人研究成果，决定选取数值天气预报中常见的五种气象要素历史数据，包括地面向下长波辐射、近地面气压、近地面空气相对湿度、地面向下短波辐射、近地面气温，结合近地面全风速的历史数据构建短期风速的预测模型。首先，采用当前科研最前沿的改进自适应噪声完备集合经验模态分解（ICEEMDAN）算法，对上述六种特征历史数据进行分解，将高频噪声和低频信号进行分离，降低时间序列数据分析的复杂度，从而提高准确性。然后，将分解结果序列按照八个时刻预测未来一个时刻的滑动窗口方法形成样本，输入由 Facebook 推出的主流时间序列预测框架 Prophet 进行统计分析，得到 Prophet 拟合分解的结果以及统计分析得出的趋势分量、累加式季节性分量以及上述分量所对应的最大边界和最小边界。这些处理后的数据进一步降低了六种特征对应时间序列的复杂度，便于神经网络的训练。最终，将所有由 Prophet 分析的时段对应的结果以及分量和原始数据一起进行基于径向基函数（RBF）的核主成分分析（KPCA），将数据升维，分解非线性相关，更进一步降低了数据复杂度。然后输入进一个由三层门控循环单元（GRU）构成的深度学习模型，采用最新科研成果 Nadam 优化器以及 Huber 损失函数训练模型，并部分应用 GELU 激活函数，输出每个时段预测的六种气象要素值，最终使用基于 RBF 的核主成分分析（KPCA）将预测结果再次升维，输入进神经网络（NN）进行修正，最终得到预测的近地面全风速值。

为了验证模型的实际效果，本文选取了甘肃中电酒泉第四风力发电有限公司附近（北

纬 40.65 度，东经 96.95 度）的 2017 年 1 月 1 日 0 时整至 2018 年 12 月 30 日 21 时整[1]间隔三小时的历史数据，使用多项指标以及对模型的预测结果进行可视化分析，对比模型各要素的优劣，从而对模型全方面综合评估，最终验证模型的优越性。

论文的结构如下：

第一章：绪论。主要讲述选题缘由，风速问题研究背景以及意义，以及对现有研究方法的综述。

第二章：数据来源及特征提取。主要讲述特征的选取原因以及数据集的制作过程，并对选取数据集的性状进行了分析，并通过统计学方法验证特征选取的正确性。

第三章：使用技术的介绍和比较。主要讲述现有主流技术，并进行技术分析对比，讲述最终选择 ICEEMDAN 分解技术、Prophet、基于 RBF 的 KPCA、GRU 网络、Nadam 优化器、Huber 损失函数以及 GELU 激活函数的原因。

第四章：模型的建立和评估。主要讲述模型的评估指标介绍、细节、训练以及调参过程，并与使用普通模型的情况进行对比。

第五章：总结与展望。对全文进行总结，指出创新点与有待进一步研究的方向。

[1]此处所述时间均为协调世界时 (UTC, Universal Time Coordinated)。

第二章 数据来源及特征选取

2.1 数据来源

本文数据来源于中国区域地面气象要素驱动数据集（1979-2018）[17]，该数据集常用于数值天气预报。该数据集包括地面向下长波辐射、地面降水率、近地面气压、近地面空气比湿、地面向下短波辐射、近地面气温共 7 个气象要素。数据时间分辨率为 3 小时，水平空间分辨率为 0.1°，为 netCDF 格式。[18]

该数据集基于世界上现有的 GLDAS 数据、普林斯顿再分析数据、GEWEX-SRB 辐射数据和 TRMM 降水数据，并整合了中国气象局的常规气象观测数据。原始数据来自卫星遥感数据、再分析数据与气象局的观测数据，并去除了非物理范围值，对于缺失数据采用 ANUSPLIN 统计插值。该数据集的精度介于气象局的观测数据和卫星遥感数据之间，优于世界上现有的再分析数据。[19]

2.2 特征选取与数据集的制作

由于气象要素之间都是紧密相关的，本文数据集基于上述中国区域地面气象要素驱动数据集（1979-2018）构建而成，通过 Python 语言 netCDF4 库 [1] 选取了数据集中的全部气象要素，并选择了甘肃中电酒泉第四风力发电有限公司附近（北纬 40.65 度，东经 96.95 度）的 2017 年 1 月 1 日 0 时整至 2018 年 12 月 30 日 21 时整 [2] 共 5832 条数据。生成数据格式为 CSV（逗号分隔值，Comma-Separated Values）文件，包含数据内容如表2.1所示：

[1]提取所用代码请见附录部分 A.1

[2]此处所述时间均为协调世界时 (UTC, Universal Time Coordinated)。

表 2.1　选取特征说明

特征	名称	单位	含义
ds	日期	UTC	数据格式为：YYYY-mm-dd HH:MM:SS
lrad	地面向下长波辐射	W/m^2	从当前时间 1.5 小时前开始的 3 小时平均值
prec	地面降水率	mm/h	从当前时间 3 小时前开始的 3 小时平均值
pres	近地面气压	Pa	近地面 (距地面 2 米处) 瞬时值
shum	近地面空气相对湿度	比值，无单位	近地面 (距地面 2 米处) 瞬时值
srad	地面向下短波辐射	W/m^2	从当前时间 1.5 小时前开始的 3 小时平均值
temp	近地面气温	K	近地面 (距地面 2 米处) 瞬时值
wind	近地面全风速	m/s	近地面 (距地面 2 米处) 瞬时值

2.3　特征数据分析

2.3.1　描述性统计分析

表 2.2　七种气象要素的描述性统计分析 [1]

指标	lrad	prec	pres	shum	srad	temp	wind
算术均值	269.5	0.009	85519.502743	0.002993	192.962834	281.441822	4.040625
标准差	64.77	0.085995	610.557534	0.002663	267.761536	13.806527	2.357598
最小值	132.3	0	83722	0.000015	0	245.629990	0.051998
25% 值	216.3	0	85049.5	0.001129	0	270.36	2.285995
50% 值	265.5	0	85518	0.001971	3	282.754990	3.529999
75% 值	321.1	0	85972	0.004012	352.5625	292.5825	5.258496
最大值	449.5	2.692501	86820	0.015863	989.5	311.25	16.23
极差	317.3	2.692501	3098	0.015848	989.5	65.620010	16.178002
四分位差	104.8	0	922.5	0.002883	352.5625	22.2225	2.972501
离散系数	0.24	9.609496	0.007139	0.889654	1.387633	0.049056	0.583474
平均离差	54.69	0.016437	508.073047	0.002045	223.512922	11.758743	1.844962
偏态	0.181	19.269338	0.002976	1.563321	1.211468	-0.237797	1.115041
峰度	-0.87	443.377569	-0.722968	2.034095	0.202964	-0.892140	1.404610

[1] 该表格数据生成代码见附录部分 A.2

使用 python 的 pandas 库自带的统计功能计算得到表2.2中统计数据。由表2.2统计数据可见，选址地（甘肃中电酒泉第四风力发电有限公司附近，北纬 40.65 度，东经 96.95 度）2017-2018 年两年全年最低气温-27.5°C，最高 38.1°C，温差较大，冷热对流明显。同时该地区降水非常稀少，空气干燥，太阳辐射强，易产生空气的稳定辐射型对流。该地区出现过的最大风力为 7 级（16.23 m/s），风力平均为 3 级（4.04 m/s），因而风速较大，且风速的方差较小，风力资源丰富且相对平稳，的确是风力发电的一个很好的选址。

2.3.2 相关性分析和显著性检验

因为传统的皮尔逊（Pearson）相关系数衡量的只是线性相关关系，并且皮尔逊相关系数对数据的要求较高，不适用于非平稳且非线性的风速数据。因而这里采用可以度量单调关系的斯皮尔曼（Spearman）相关系数。使用 python 的 scipy 库相关方法计算得到表2.3中统计数据。

表 2.3　风速与其他六种气象要素的斯皮尔曼相关性分析 [1]

wind	相关系数	显著水平
lrad	0.032350777005597936**	0.013486168347444174
prec	0.00787600133073642	0.5476061053614518
pres	0.07469086091232147**	$1.1250462841430117 \times 10^{-8}$
shum	−0.08556207603752566**	$5.958774799092212 \times 10^{-11}$
srad	0.246264995629444**	$2.650405346230128 \times 10^{-81}$
temp	0.05671501296224254**	$1.4659365394277926 \times 10^{-5}$

[1] 该表格数据生成代码见附录部分 A.3

** 表示在 0.01 级别（双尾），相关性显著

由表2.3所示，排除由于选址地区降水数据过于稀少，导致地面降水率与其相关性不大的因素，风速与其他剩余五种气象要素的相关系数都较大，有较强的相关性，并且相关性都在 99% 的置信水平下通过检验，相关性极其显著。其中与风速相关性最强的是地面向下短波辐射。

最终根据上述过程中的数据分析，去除掉地面降水率这一特征，保留剩余的地面向下长波辐射、近地面气压、近地面空气相对湿度、地面向下短波辐射、近地面气温共计五种特征。

第三章　使用技术的介绍和比较

3.1　基于 EMD 的信号分解技术

3.1.1　EMD

经验模态分解（EMD，Empirical Mode Decomposition）是一种适用于处理非平稳非线性时间序列的自适应时空分析方法，由 Huang, Norden E 等人于 1998 年提出 [20]。EMD 在不离开时域的情况下将序列划分为有限个本征模函数（IMF，Intrinsic Mode Function）和残差的操作。分解出的 IMF 可以提供一个时间序列中包含的各种信号的特征。与傅里叶变换和小波分解等类似，EMD 分解不是基于物理性质。但是，EMD 依据信号自身的时间尺度特征对信号没有任何先验主观标准选择的进行自动分解，因而无需像傅里叶变换与小波分解一样设定谐波基函数或小波基函数。

在时域范围内具有以下条件特征的时间序列，可以使用 EMD 分解：

- 局部的时域特性仅由极值点间的时间尺度唯一确定。
- 至少存在一个极大值点和一个极小值点。
- 若存在拐点但无极值点，可通过多次微分求得极值，然后积分计算分解结果。

分解出的本征模函数在时域范围内满足如下两个性质：

- 极值点和过零点的数目相等或相差一个。
- 上包络线（极大值点的包络线）和下包络线（极小值点的包络线）均值为 0。

EMD 分解步骤如下：

1. 找出待分解时间序列 $X(t)$ 的所有极大值点，并将所有极大值点用三次样条插值函数拟合形成原数据的上包络线。
2. 同理，找出待分解时间序列 $X(t)$ 的所有极小值点，并将所有极小值点用三次样条插值函数拟合形成数据的下包络线。
3. 设上包络线和下包络线的均值函数为 $M(t)$，设 $H(t)=X(t)-M(t)$，得到新的时间序列 $H(t)$。如果 $H(t)$ 不满足本征模函数的两个性质，则对 $H(t)$ 重复步骤 1-3，否则得到一个 IMF 分量 $H(t)=IMF_i(t)$。
4. 对 $M(t)$ 重复步骤 1-3，直到 $M(t)$ 不满足使用 EMD 分解的条件，则 $R(t)=M(t)$，R(t) 为残差。

由上述推导过程可得：

$$X(t)=\sum_{i=1}^{n} IMF_i(t)+R(t)$$

3.1.2 EEMD

集成经验模态分解（EEMD，Ensemble Empirical Mode Decomposition）是一种噪声辅助数据分析方法，包含了用于筛选的白噪声附加时间序列。EEMD 由 Wu, Zhaohua 等人提出 [21]。他们的研究发现，在 EMD 分解的 IMF 分量筛选过程中，添加一定的白噪声可以平滑极值点的分布，从而让分解的结果更加健壮。使分解出的时间序列在适当的本征模函数（IMF）中进行比较，从而能够让分解过程考量到尽多可能的解并进行筛选。由于 EMD 是一种时空分析方法，因此白噪声可以通过足够多的平均迭代抵消，从而在此平均化过程中唯一留存下来的部分就是更真实且更具物理意义的时间序列。

EEMD 分解步骤如下：

1. 对待分解时间序列 $X(t)$ 添加一个随机白噪声序列 $N(t)$，得到新序列 $K(t)$。
2. 对 $K(t)$ 进行 EMD 分解。
3. 对步骤 1-2 重复 n 次，将得到的共计 n 组的每个 IMF 分量和 R 按组进行系综平均（Ensemble Average）。
4. 得到系综平均后的 IMF 分量和 R，公式同 EMD 分解。

3.1.3 CEEMDAN

自适应噪声完备集合经验模态分解（CEEMDAN，Improved Complete Ensemble Empirical Mode Decomposition with Adaptive Noise) 由 Torres 等人提出 [22]。它是 EMD 的改进版本，同时又借鉴了 EEMD 的思想，提供了对原始信号的精确重建和对本征模函数（IMF）的更好的频谱分离。

CEEMDAN 分解步骤如下：

1. 对待分解的时间序列 $X(t)$ 添加一个幅值为 r 的白噪声序列 $N(t)$，得到新序列 $K(t)$。
2. 对 $K(t)$ 进行一次 EMD 分解迭代，得到一个 IMF 分量。
3. 对步骤 1-2 重复 n 次，将得到的共计 n 组的 IMF 进行系综平均（Ensemble Average），得到 $IMF_i(t)$ 用 $X(t)-IMF_i(t)$ 作为新序列，返回步骤 1 迭代 j 次。
4. 最终不能进行 EMD 分解时 $X(t)-IMF_i(t)$ 得到残差 R，公式关系同 EMD。

CEEMDAN 相较于 EEMD 方法有如下优点：

- EEMD 方法需要通过足够多的平均迭代抵消白噪声，因而将所有 IMF 分量和残差 R 直接相加复原出的原信号并不准确。而 CEEMDAN 在较小的平均次数下就可以有很好的可复原性。
- 相较于 EEMD，CEEMDAN 无需计算过多的平均值，因而分解的性能要更高。
- EEMD 分解一般会出现多个无意义的低幅值低频伪 IMF 分量，CEEMDAN 可以减少该种情况出现的频率。

3.1.4 ICEEMDAN

ICEEMDAN 由 Colominas 等人提出 [23]，其为 CEEMDAN 的改进版本。ICEEMDAN 将上述步骤 3 的重复迭代过程中的第 x 次迭代时使用的噪声更改为原第一次添加噪声的对噪声进行 EMD 分解的第 x 个 IMF 分量乘以添加噪声相对于待分解时间序列的信噪比，再除以白噪声的标准差。

ICEEMDAN 分解相对于 CEEMDAN 进一步降低了出现多个无意义的低幅值低频伪 IMF 分量的频率。

由于 ICEEMDAN 是基于 EMD 的信号分解技术的集大成者，因而本文最终决定选用 ICEEMDAN。

3.2 基于 Prophet 框架的时间序列预测方法

Prophet [24] 是由 Facebook 于 2018 年提出的一种时间序列预测算法框架。其核心部分使用类似于 SARIMA 的方法进行时间序列预测，可以学习年、周和日、年的季节性以及假日效应等非线性趋势，最适用于具有强烈季节性影响的时间序列，可以通过 R 和 Python 实现调用。Prophet 的预测速度快，不同于 SARIMA 需要手动对时间序列进行处理判断以及定阶，Prophet 提供了完全自动化的时间序列预测功能，因而无需手动操作即可对杂乱的数据进行合理预测。其对异常值、缺失数据和时间序列中的剧烈变化也都具有十分强大的健壮性，并且为用户提供了许多超参数，支持通过交叉验证进行超参数调优。

Prophet 的算法模型为：

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_{(t)}$$

该公式中 g(t) 表示的是趋势分量，代表时间序列数据在非周期性规律中的变化趋势；s(t) 表示周和日、年的季节性分量；h(t) 表示节假日项，此处和风速预测无关；ε_t 为剩余项，表示噪声分量，其服从高斯白噪声分布。

3.3 基于 PCA 的降维方法

3.3.1 主成分分析

主成分分析（PCA，Principal Component Analysis）是一种基于线性变换降低数据集的维度，从而达到简化数据集目的的方法。其利用正交变换对一系列可能的相关变量的观测值进行线性变换，从而将其投影为一系列线性不相关的变量特征，并同时保留对彼此差异贡献最大的特征。这种变换这是通过忽略高维成分并保留低维成分并来实现的。

假设待降为的原始数据有 m 条，每条数据为 n 维，组成了 n 行 m 列的矩阵 X，则 PCA 的算法步骤如下：

- 首先将 X 的每一行进行零均值化（减去这一行的均值）。

- 求解协方差矩阵 $C=\frac{1}{m}XX^{\mathsf{T}}$，以及其特征值及对应的特征向量。
- 将协方差矩阵 C 的特征向量按对应特征值大小从前到后按行排列成矩阵，取前 k 行组成新矩阵 P。
- $Y=PX$ 即为降维到 k 维后的数据。

3.3.2 核主成分分析

PCA 可以很好的解除线性相关，但是对于非线性的高阶相关性，PCA 无法处理 [25]。核主成分分析（KPCA）是一种基于核函数的主成分分析方法，其类似于 SVR（支持向量回归），通过核函数将非线性相关变换到更高维度，使其转换为线性相关。因而，KPCA 方法可以用于解决非线性的高阶相关性问题，适用于非平稳非线性关系较强的天气要素数据。

本文核函数使用径向基函数（高斯核函数，RBF 核，Radial Basis Function Kernel），其表达式为：

$$K(\mathbf{x},\mathbf{y})=e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$$

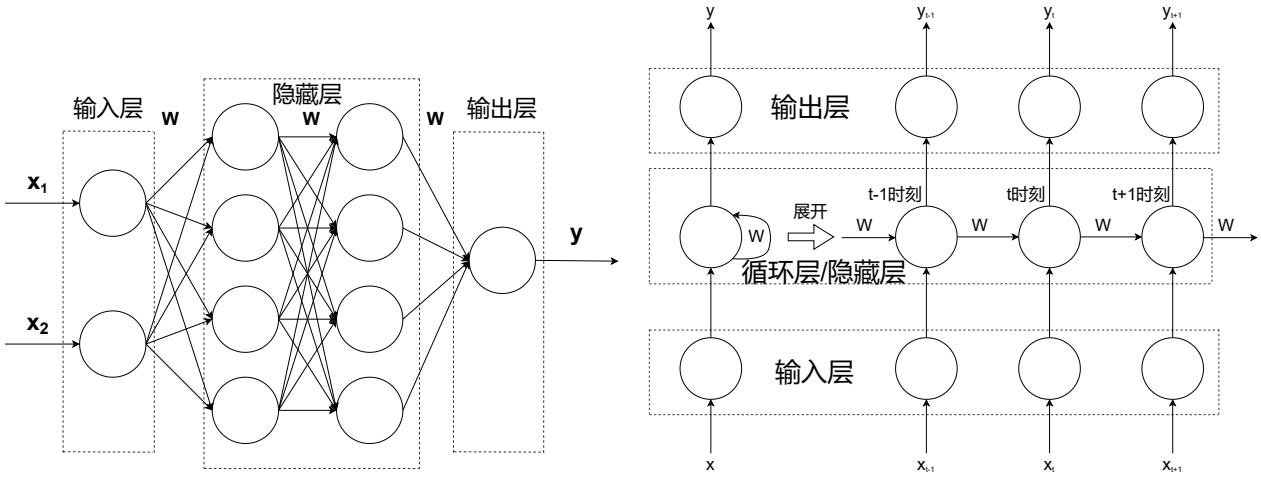
与其他核函数相比，选择 RBF 核函数的原因是：

1. RBF 为非线性映射核函数，和风速数据中较多的非线性特征契合，并且线性核函数也只是 RBF 的一个特例，因而不使用线性核函数。
2. 多项式核函数需要计算内积，容易发生数值溢出的问题。
3. Sigmoid 核函数 $K(\mathbf{x},\mathbf{y})=\tanh(b\mathbf{x}^T\mathbf{y}-c)$ 需要确定超参数 b 和 c，比较麻烦，无法实现自动化 KPCA。

3.4 基于 RNN 的深度学习神经网络模型

3.4.1 经典循环神经网络原理

循环神经网络（RNN，Recurrent Neural Network）是一种深度学习模型。经典的神经网络 [26]，多用于图像处理。由于其时序无关，采用了层状传递结构，模型的内部前一层神经元直接传递给后一层神经元，这样做模型由于无记忆功能，无法处理时序数据中的历史相关性，因而需要循环层内部传递数据来处理这种时序数据。图3.1展示了两种神经网络的结构区别。



(a) 经典 BP 神经网络　　(b) 循环神经网络

图 3.1　两种神经网络结构示意图

假设激活函数为 sigmoid，RNN 神经网络的每个神经元都是通过输入值和上一层的记忆值同时乘对应权重并与偏差相加，最终再通过 sigmoid 激活函数将结果压缩在 0 和 1 之间得出输出。上述过程得到如下所示公式：

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_z)$$

公式中 W 代表权重，b_z 是偏差，x_t 是 t 时刻的神经元输入值，h_{t-1} 是 t-1 时刻神经元的记忆输出值，W 和 U 为对应的值的权重，h_t 是当前神经元的输出值。训练时对每个神经元采用反向传播 (BP) 算法，使用训练集中的数据，通过指定学习优化器和损失函数来迭代计算权重和偏差，从而使误差的最小化。

由循环神经网络的结构和原理也可以看到，由于其在训练反向传播时权重产生的误差都会累加迭代，因而很容易误差累加数值过大，造成梯度爆炸。同时，由 sigmoid 激活函数的性质可知，当值较大时，函数趋于 1，并变得平缓，此时进行反向传播会造成整体值过小，造成梯度消失的问题。另外，循环神经网络的训练过程中每个神经元权重都是直接记忆学习临近时刻的，因而对可能会存在对类似季节性效应的长期关系学习不佳。

3.4.2　GRU 原理

上述 RNN 的三个问题中，梯度爆炸可以采用阈值范围对梯度进行裁剪就可以解决，但是梯度消失和长期依赖关系的问题则必须要改变模型的结构。LSTM 是首先被提出的解决方案 [27]，GRU 则是相对于 LSTM 更简单且更优异的方案。

GRU 是由 Cho 等人于 2014 年提出的一种类似于 LSTM 的神经网络模型 [28]。本文之所以选择 GRU 而非 LSTM，是因为相较于 LSTM 的结构有三个门控单元输入门、遗忘门和输出门，GRU 只有两个，分别为更新门和重置门。并且 GRU 的单个神经元中并不会保

留内部记忆，因而 GRU 的参数更少，复杂度更低，收敛速度更快，并且 GRU 和 LSTM 所能达到的训练以及预测效果同样出色，因而权衡效果和成本而言 GRU 要更高 [29]。

GRU 的工作原理如下：

首先更新门确定有多少过去的信息（来自之前的时间步长）需要传递到未来。计算时间 t 的更新门 z_t 的值，执行计算公式如下：

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

该公式和标准 RNN 的计算公式类似。

重置门决定要忘记多少过去的信息，其公式与更新门的公式相同：

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

然后执行以下计算公式：

$$h'_t = tanh(Wx_t + r_t \odot Uh_{t-1})$$

该步骤中重置门和过去的神经元输出进行元素积，确定遗忘掉多少过去的信息，并和当前的输入值进行整理相加。在此步中，若选择使 r_t（重置门）的值接近 0，则可以遗忘掉过去的大部分信息。

最后一步，确定当前步骤需要记忆的信息：

$$h_t = z_t \odot r_t + (1 - z_t) \odot h'_t$$

该模型可以选择使 z_t（更新门）的值接近 1，从而保留大部分过去的信息。并且由于此时 $1 - z_t$ 接近 0，因而将忽略对当前内容的记忆。

从上述推导步骤可以得出 GRU 模型解决长期记忆问题的方法。同时，接近 1 和接近 0 的同时出现也解决了梯度消失的问题。

从上述推导过程还可以得出，如果选择将重置门设置为 1，更新门设置为 0，那么 GRU 将变为标准的 RNN 模型。

3.4.3 神经网络模型激活函数

激活函数将神经网络模型中神经单元的输出映射到一个范围，从而起到控制输出数据值域的作用。

a. sigmoid

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

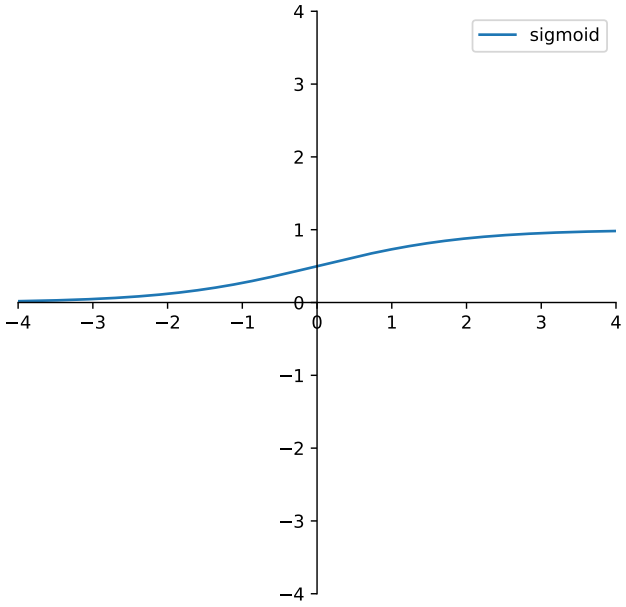


图 3.2　sigmoid 函数图像

如图3.2所示，sigmoid 函数为单调递增函数，在 $-\infty$ 上趋近于 0，在 $+\infty$ 上趋近于 1。

b.　tanh

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

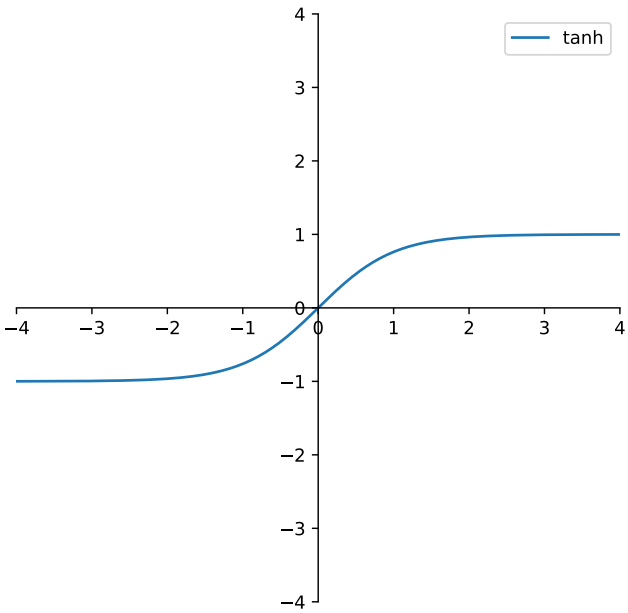


图 3.3　tanh 函数图像

如图3.3所示，tanh 函数为单调递增函数，在 $-\infty$ 上趋近于-1，在 $+\infty$ 上趋近于 1，且过原点。其实质上是 sigmoid 函数的缩放移动版本。

c.　ReLU 和 GELU

$$ReLU = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

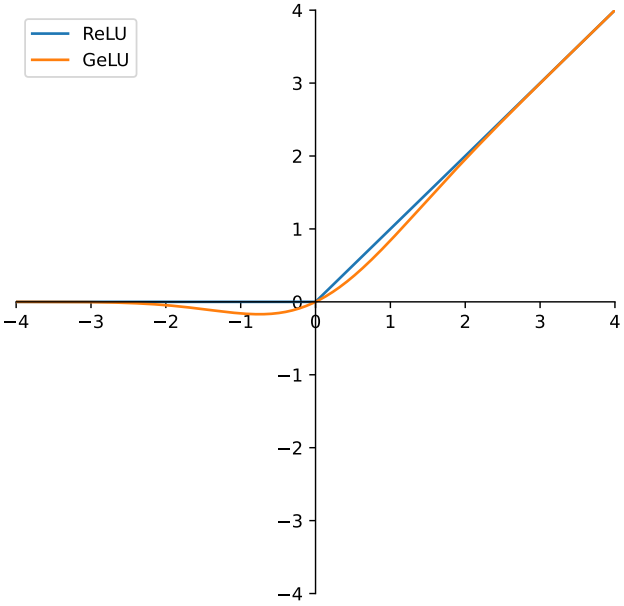


图 3.4　ReLU 和 GELU 函数图像

如图3.4所示，尽管整流线性单元（ReLU，Rectified Linear Unit）在 x 取负值时为 0，符合近地面全风速数据不为负的特点，但是由于 ReLU 是线性激活函数，而风速数据具有较强的非线性特征，因而用于风速数据效果不是很理想。

高斯误差线性单元 (GELU，Gaussian Error Linear Unit）是 Hendrycks, Dan 等人于 2016 提出的 [30] 一种高性能的神经网络激活函数。GELU 激活函数表达式是 $x\Phi(x)$，其中 $\Phi(x)$ 是标准高斯累积分布函数。GELU 是非线性激活函数，其对输入值进行高斯加权处理，能带给训练任务更好的精度和拟合效果。

3.4.4　神经网络模型优化器

本部分内容参考了 Ruder, Sebastian 的梯度下降优化算法综述 [31]。

a.　SGD

随机梯度下降（SGD，Stochastic Gradient Descent）是深度学习神经网络的训练基础算法，它是一种比较简单的梯度下降算法。假设目标神经网络函数为 $J(\theta)$，神经网络的参数值为 θ。则 SGD 根据固定的学习率 η，通过在目标函数梯度的相反方向上 $\nabla_\theta J(\theta)$ 更新参数 θ 来搜寻神经网络的参数，使得目标神经网络达到局部最优点。

SGD 的更新公式为：

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

b.　Momentum

Momentum 是 Qian, Ning 研究出的 [32] 一种有助于在梯度下降的方向加速 SGD 并抑制 SGD 振荡的方法。它通过添加动量项比值 γ，将过去时间步的更新向量到添加到当前的更新向量，来实现这一目的。

Momentum 的更新公式为：

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$

c. NAG

涅斯捷罗夫加速梯度下降（NAG，Nesterov Accelerated Gradient）[33] 基于 Momentum，通过 $\theta - \gamma v_{t-1}$ 项使其能够预知未来的更新方向，从而使得更新更加稳定而不至于一直遵循梯度更新的惯性。这种预期性的更新可以防止梯度更新得太快，从而提高响应能力。NAG 显著提高了 RNN 在许多训练任务中的性能 [34]。

NAG 的更新公式为：

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$
$$\theta = \theta - v_t$$

d. Adam

Adam [35] 是一种基于 RMSProp 的优化器，它能够自动化调整学习率，对与频繁出现的特征相关的参数执行较小的更新（低学习率），对与不常见特征相关的参数执行较大的更新（高学习率），同时还能适应稀疏数据，克服学习率急剧下降的问题。其本质上为带 Momentum 动量项的 RMSProp。

e. Nadam

Nadam [36] 向 Adam 优化器中融合了 NAG 的思想，添加了 Nesterov 动量，从而使其获得了 NAG 的能够预知未来的更新方向的优点，提高其在 RNN 训练任务中的性能，最终使用 Nadam 能够取得比 Adam 更好的效果。

3.4.5 神经网络模型损失函数

a. 均方损失函数（MSE）

$$MSE = \sum_{t=1}^{N} (\hat{y}(t) - y(t))^2$$

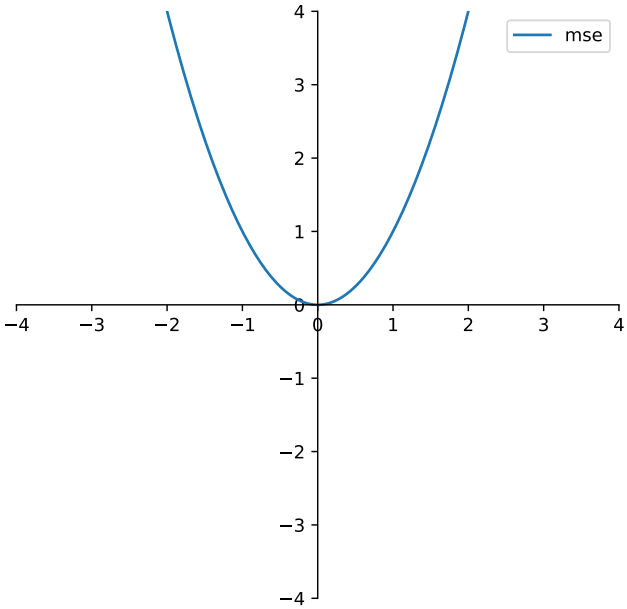


图 3.5　mse 函数图像

MSE 是平方损失函数，如图3.5所示，其光滑连续且可导，并且随着误差的减小，梯度也在减小，从而利于函数的收敛。因而 MSE 适合于对其使用梯度下降算法。但是，由于 MSE 的梯度会随着误差的增大而增大，如果样本中存在较多的异常点，MSE 会给这些异常点平方倍的权重，从而牺牲正常点的回归预测效果。

b.　平均绝对损失函数（MAE）

$$MAE = \sum_{t=1}^{N} |\hat{y}(t) - y(t)|$$

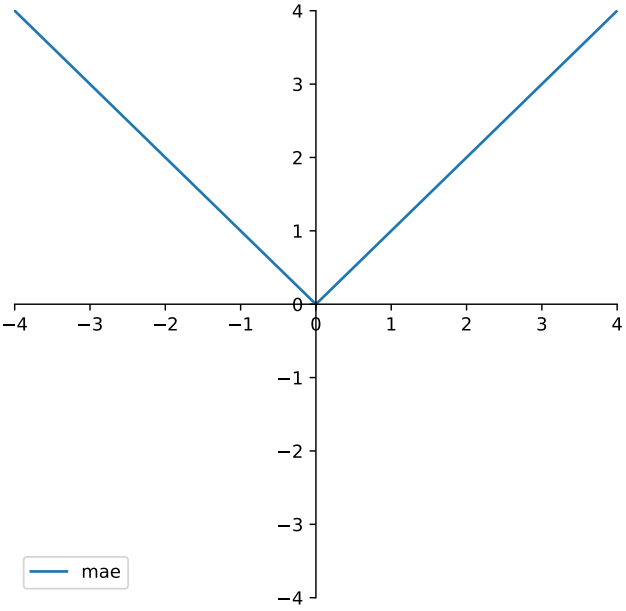


图 3.6　mae 函数图像

如图3.6所示，MAE 是连续且在非零点处可导的，但是由于其是线性损失函数，梯度始终保持不变。因而在优化器学习率保持不变的情况下，MAE 不利于函数的收敛。但是，由于 MAE 将异常点和正常点同等看待，因而克服了 MSE 的缺点。

c. Huber 损失函数

$$Huber = \begin{cases} \frac{1}{2}(y(t)-\hat{y}(t))^2, & |(y-\hat{y})| < \delta \\ ((y(t)-\hat{y}(t))-\frac{1}{2}\delta)\delta, & |(y-\hat{y})| \geq \delta \end{cases}$$

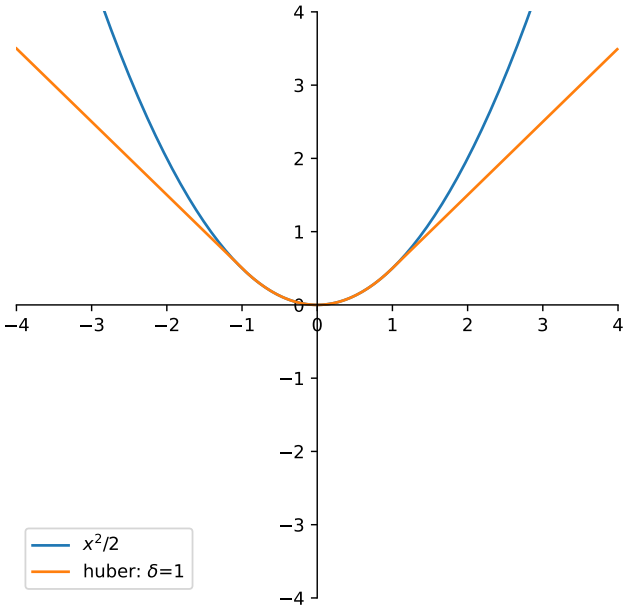


图 3.7　huber 函数图像

Huber 损失函数 [37] 包含了一个超参数 δ。当预测偏差小于 δ 时，它采用 MSE 变换形式的平方损失函数；当预测偏差大于等于 δ 时，采用 MAE 变换形式的线性损失函数，如图3.7所示。因而，Huber 同时综合了 MSE 和 MAE 的优点并同时克服了他们的缺点。

Huber 光滑连续且可导，当误差达到临界点 δ 时随着误差的减小，梯度也在减小，对于异常点的误差超出临界点 δ 时，不会给这些异常点很大的权重，从而同时达到一个很好的收敛和预测效果。

第四章　模型的建立和评估

4.1　六种特征数据的 ICEEMDAN 分解

下述 ICEEMDAN 分解以及时域转频域和图片生成代码请见附录部分 A.4。

这里使用 python 的 pyEMD 库实现 ICCEMDAN 分解[1]，并使用 numpy 库相关方法进行傅里叶变换生成幅频，最终使用 Matplotlib 进行可视化作图。

图中第一行为原始数据，最后一行为 ICEEMDAN 分解后得到的残差，中间的行为 ICEEMDAN 分解后得到的本征模函数（IMF）分量。其中，每行左侧为原始得到的时域数据，横坐标范围为 [0,5839]，单位为三小时（3hr）。右侧为经过傅里叶变换后得到的频域数据，横坐标范围为 [0,2919]，单位为赫兹（HZ）。

1. 地面向下长波辐射

[1]尽管 pyEMD 库中调用方法和类名为 CEEMDAN，但根据官方文档，其内部实现时是采用的改进的 CEEMDAN，即 ICEEMDAN 发表论文中所述算法。

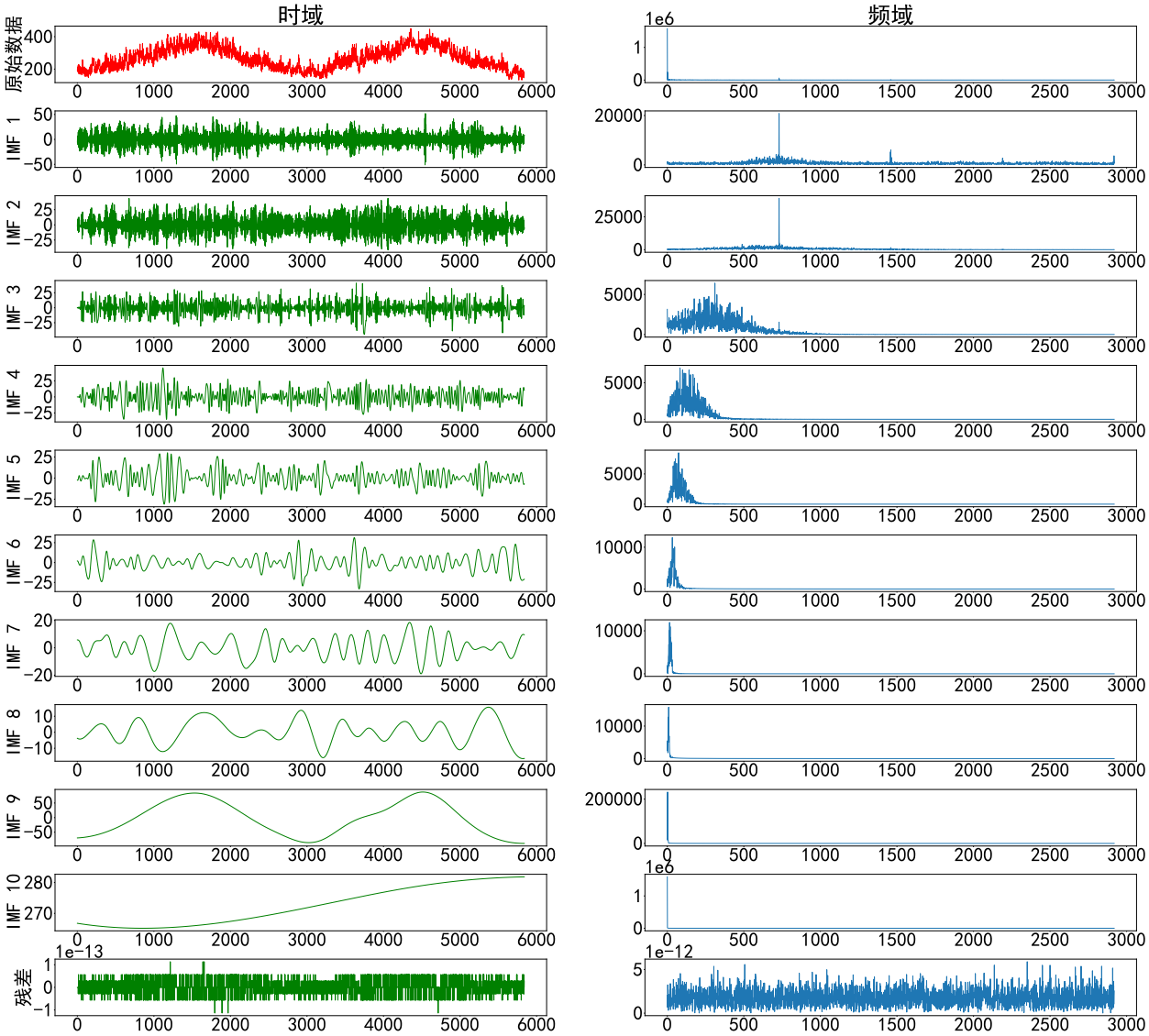


图 4.1　地面向下长波辐射的原始数据、ICEEMDAN 分解 IMF 分量、残差（左）及对应幅频图（右）

从图4.1中可以看到，原始信号中高频噪声含量较高。ICEEMDAN 将地面向下长波辐射的原始时间序列分解为 10 个本征模函数（IMF）分量，分解效果显著。由幅频可以看出，高频噪声主要集中在 IMF1 分量和 IMF2 分量中，集中于 500HZ-3000HZ 处。

2. 近地面气压

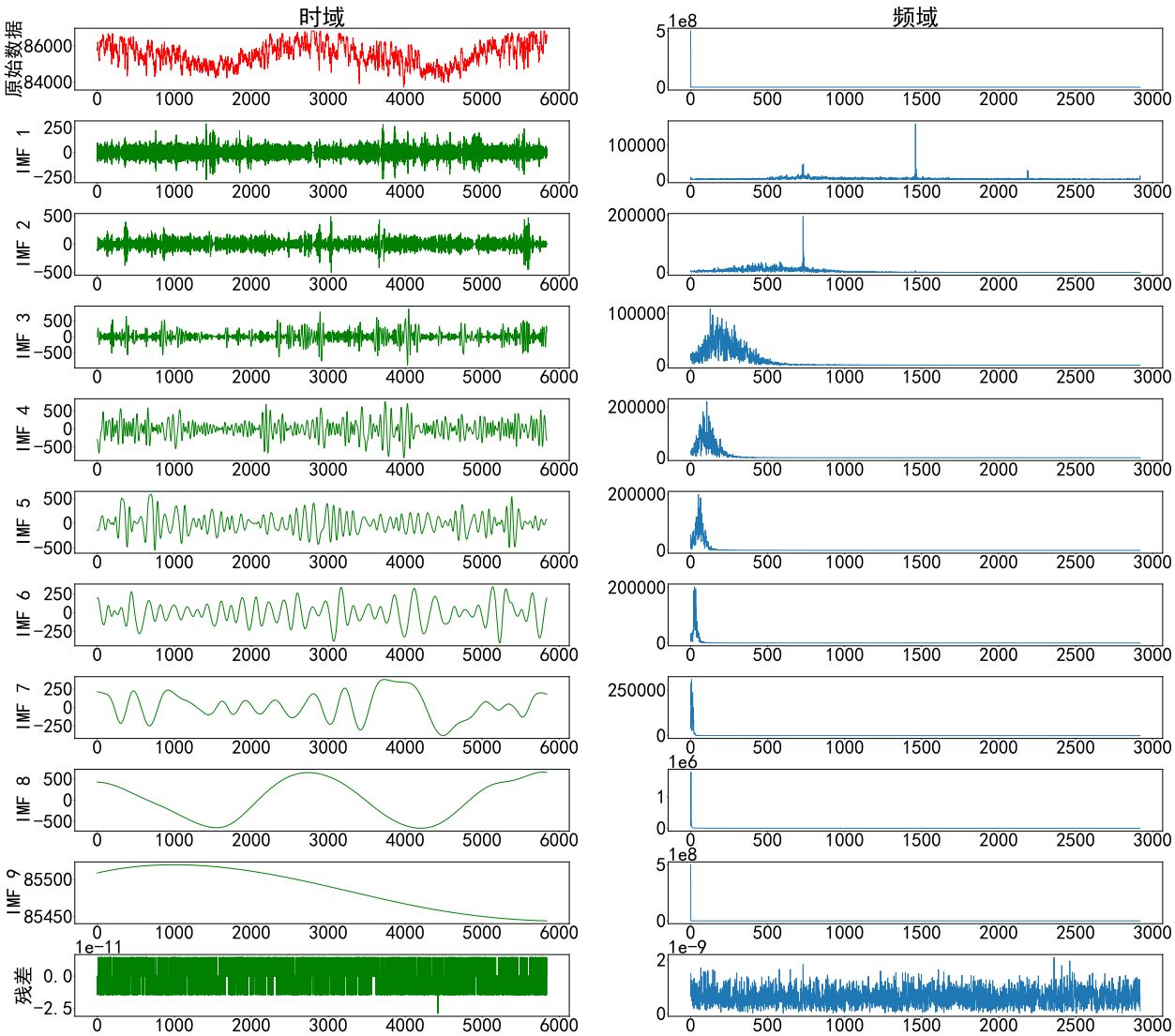


图 4.2　近地面气压的原始数据、ICEEMDAN 分解 IMF 分量、残差（左）及对应幅频图（右）

从图4.2中可以看到，原始信号中高频噪声含量同样较高。ICEEMDAN 将近地面气压的原始时间序列分解为 9 个本征模函数（IMF）分量，分解效果显著。由幅频可以看出，高频噪声主要集中在 IMF1 和 IMF2 分量中，集中于 500HZ-2300HZ 处。

3. 近地面空气相对湿度

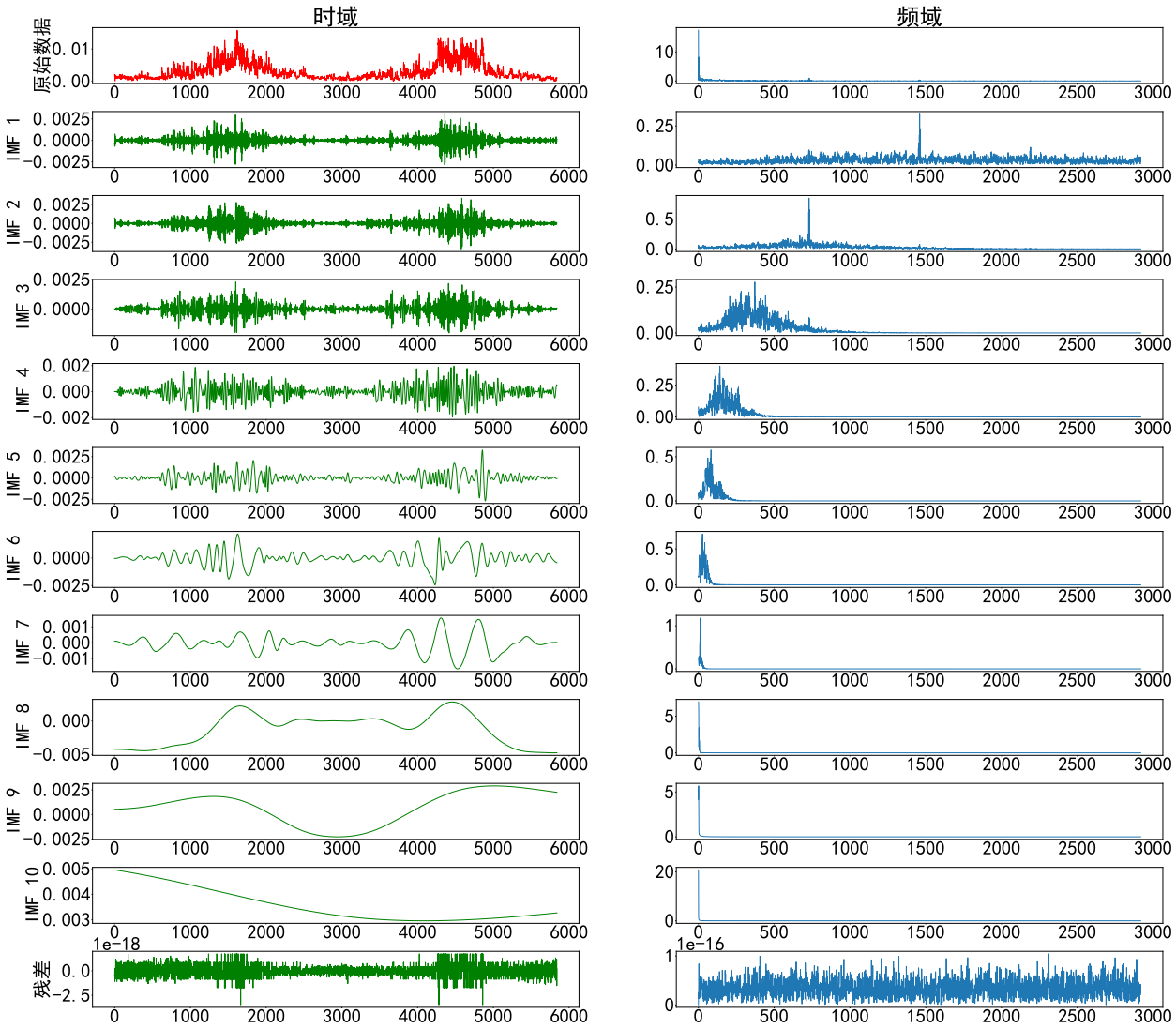


图 4.3　近地面空气相对湿度的原始数据、ICEEMDAN 分解 IMF 分量、残差（左）及对应幅频图（右）

从图4.3中可以看到，原始信号中高频噪声含量也较高。ICEEMDAN 将近地面空气相对湿度的原始时间序列分解为 10 个本征模函数（IMF）分量，分解效果较好。由幅频可以看出，高频噪声主要集中在 IMF1 和 IMF2 分量中，集中于 750HZ-3000HZ 处。

4. 地面向下短波辐射

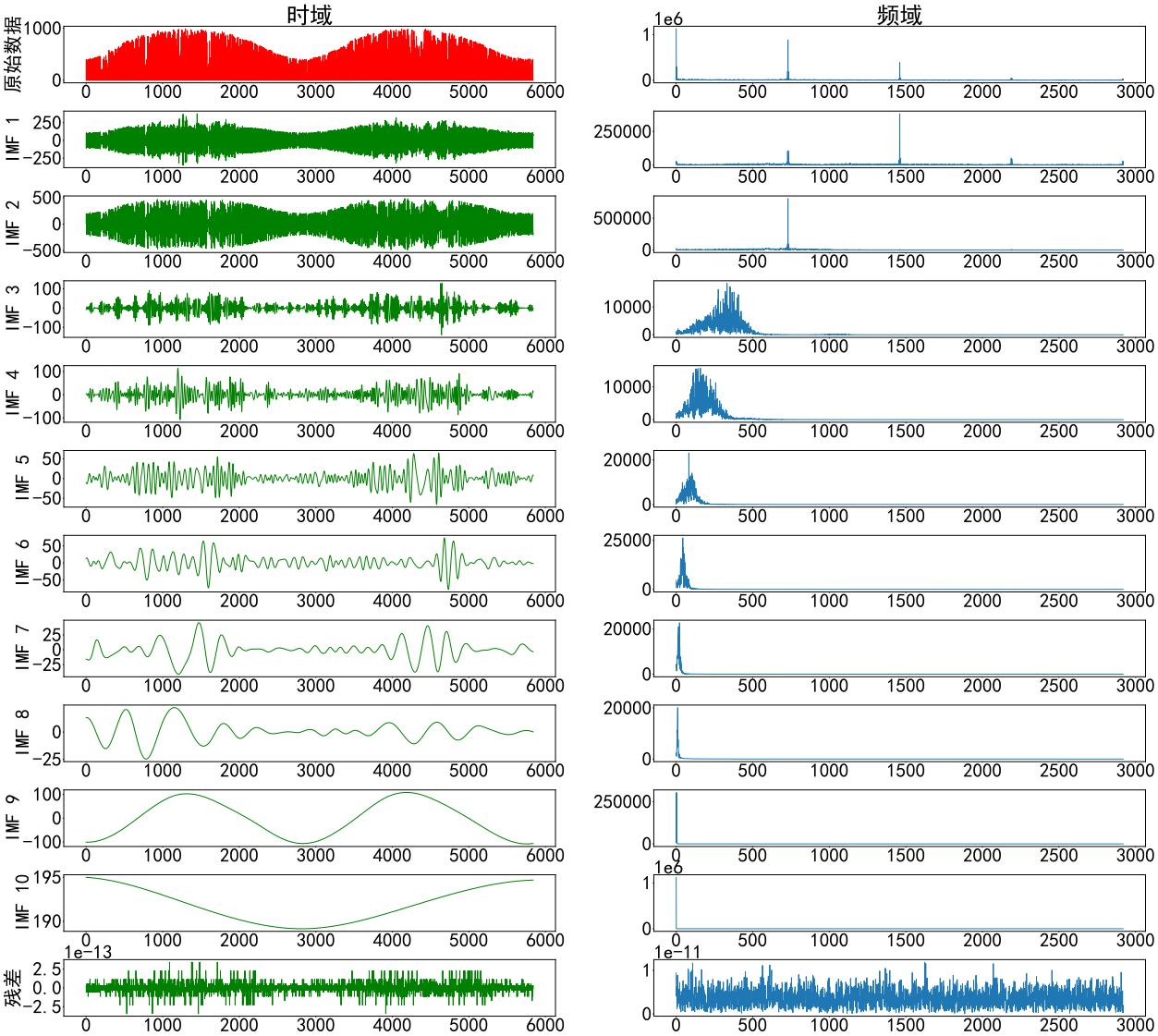


图 4.4　地面向下短波辐射的原始数据、ICEEMDAN 分解 IMF 分量、残差（左）及对应幅频图（右）

从图4.4中可以看到，原始信号中高频噪声含量也较高。ICEEMDAN 将地面向下短波辐射的原始时间序列分解为 10 个本征模函数（IMF）分量，分解效果十分显著。由幅频可以看出，高频噪声主要集中在 IMF1 和 IMF2 分量中，集中于 750HZ-2200HZ 处。

5. 近地面气温

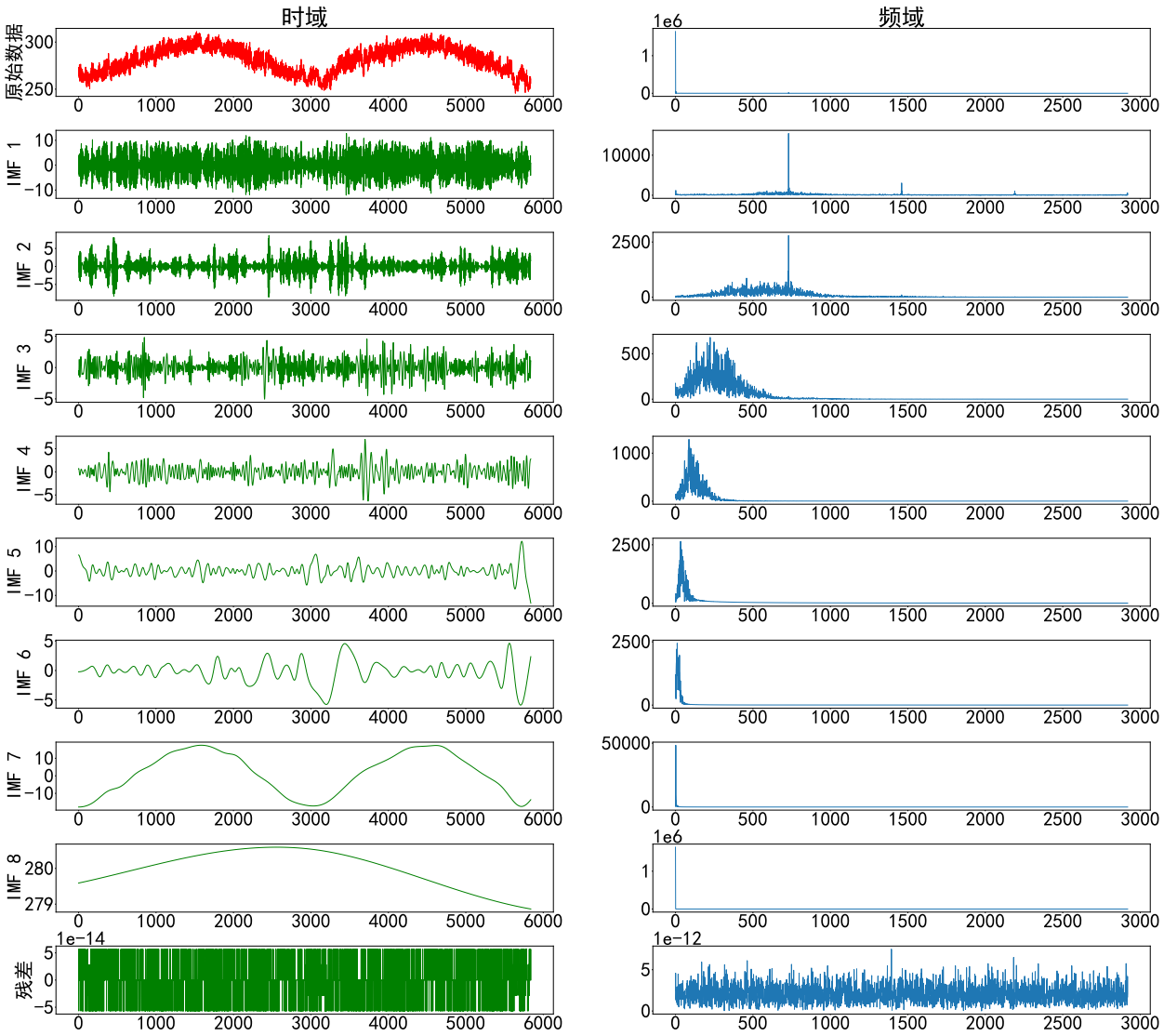


图 4.5　近地面气温的原始数据、ICEEMDAN 分解 IMF 分量、残差（左）及对应幅频图（右）

从图4.5中可以看到，原始信号中高频噪声含量同样较高。ICEEMDAN 将近地面气温的原始时间序列分解为 8 个本征模函数（IMF）分量，分解效果显著。由幅频可以看出，高频噪声主要集中在 IMF1 和 IMF2 分量中，集中于 500HZ-2200HZ 处。

6. 近地面全风速

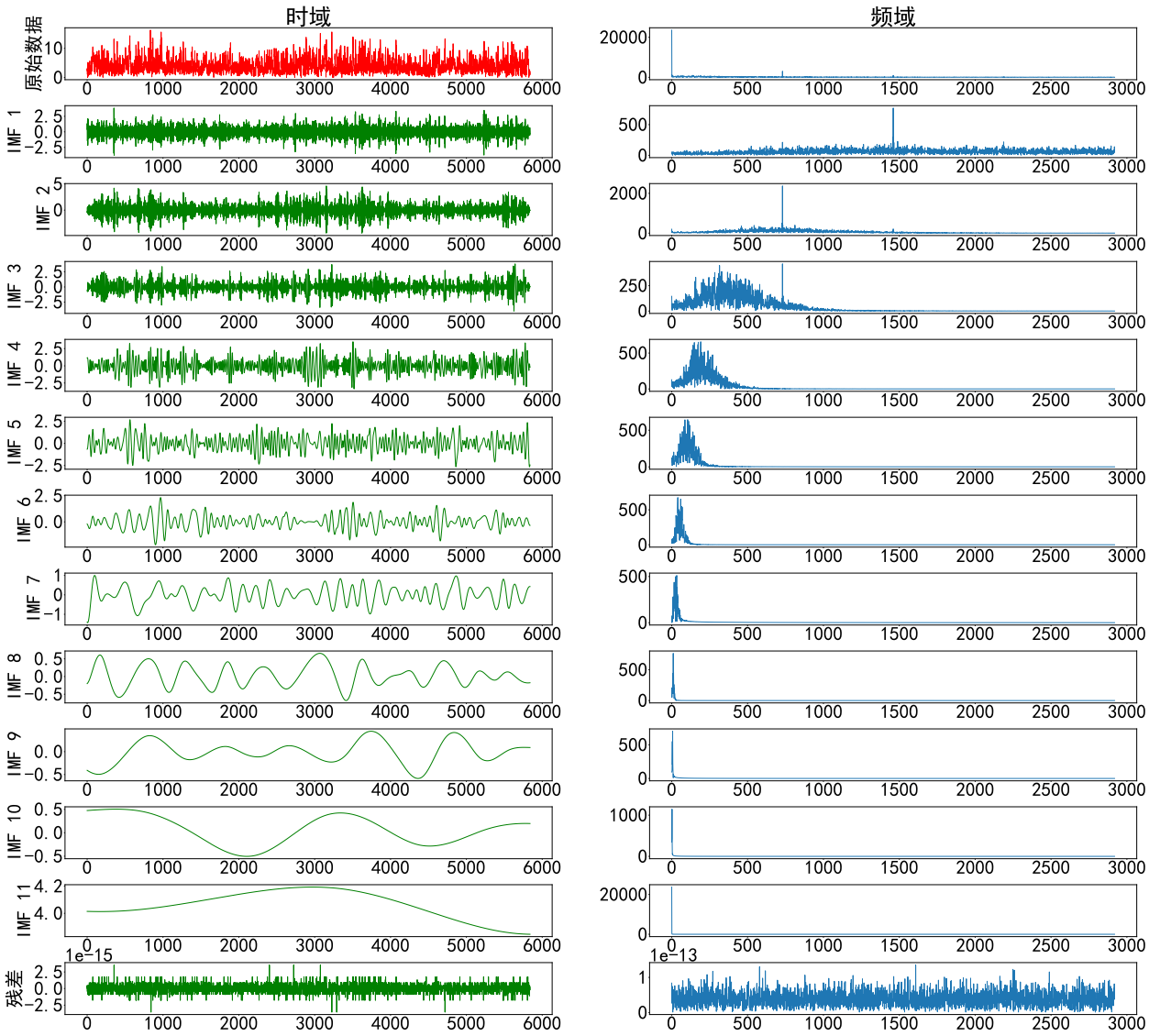


图 4.6　近地面全风速的原始数据、ICEEMDAN 分解 IMF 分量、残差（左）及对应幅频图（右）

从图4.6中可以看到，原始信号中高频噪声含量也较高。ICEEMDAN 将近地面全风速的原始时间序列分解为 11 个本征模函数（IMF）分量，分解效果显著。由幅频可以看出，高频噪声主要集中在 IMF1 和 IMF2 分量中，集中于 500HZ-3000HZ 处。

将上述信号中高频噪声对应的 IMF 分量去除，剩余的低频信号 IMF 分量相加，即可得到经过 ICCEMDAN 分解的时间序列信号。

4.2　搜寻 Prophet 模型最优超参数

将时间序列数据作为输入，首先使用 Prophet 自带的实现方法，对数据进行交叉验证以评估 12 小时的预测性能，为超参数调优提供依据。首先对训练集和验证集进行六四分，共 730 天的数据选择前 438 天（3504 条数据）作为初始训练数据。对于后 295 天（2324 条数据），每隔 1 天对未来 12 小时的情况进行一次预测，共计验证 291 次，搜寻 Prophet 的最优

超参数，从而使得平均绝对误差百分比（MAPE）值最小化。使用的代码见附录部分 A.5。最终得到对于风速序列的最优参数如表4.1所示：

表 4.1 风速历史时间序列数据 Prophet 模型的最优超参数

超参数名	值
changepoint_prior_scale	1.0
seasonality_prior_scale	0.1
seasonality_mode	additive
changepoint_range	1

4.3　基于风速历史时间序列数据的 ICEEDMAN-GRU 模型

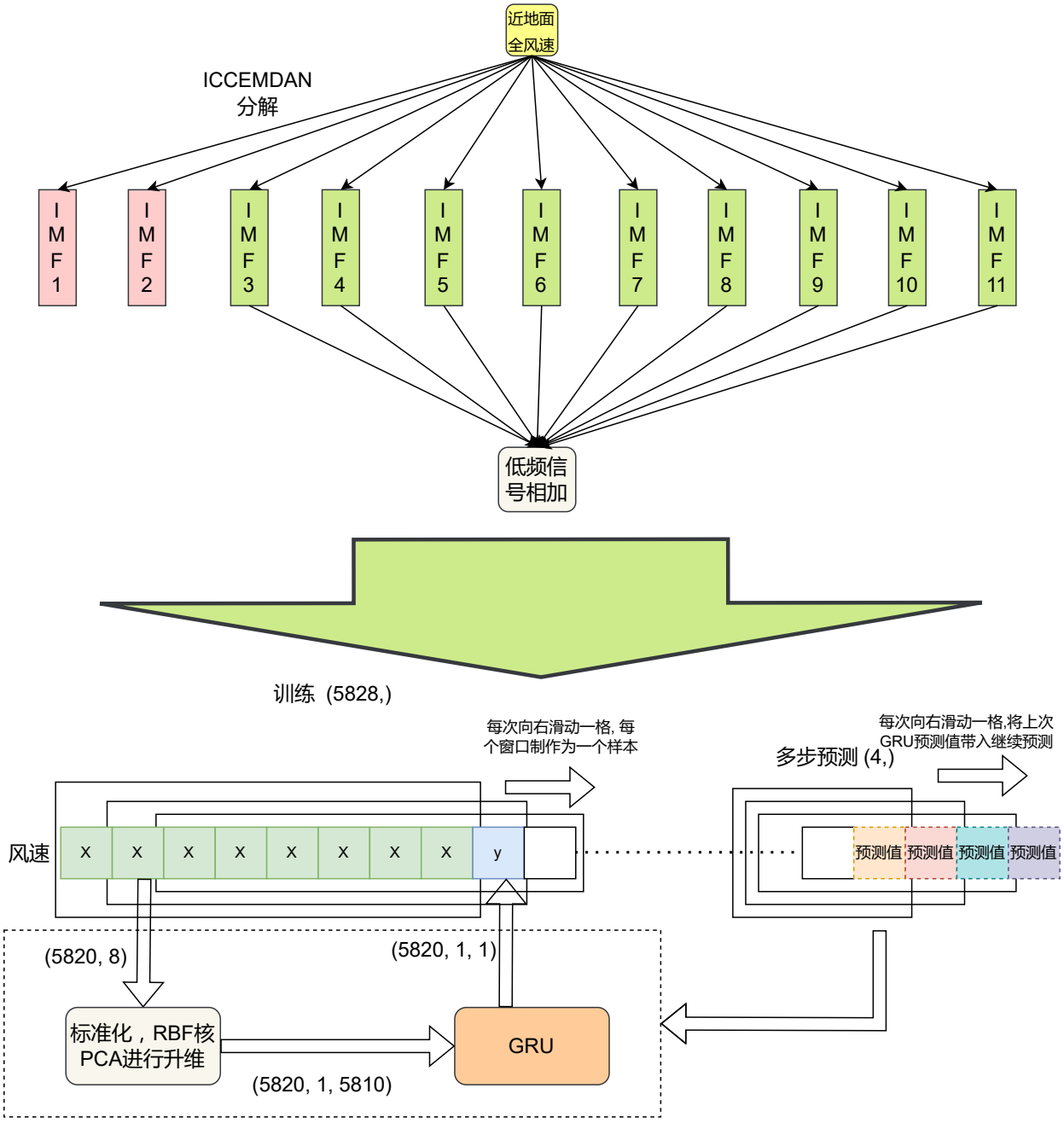


图 4.7　基于风速历史时间序列数据的 ICEEDMAN-GRU 模型结构图

图4.7展示了基于风速历史时间序列数据的 ICEEDMAN-GRU 模型结构示意。该模型使用了历史风速时间序列中的 5828 条数据点，以 8 个连续时间序列数据点（24 小时，1 天）为一个时间窗口制作为一个训练样本。每个时间窗口中的数据点整体为一个输入，每个时间窗口中的输出为该输入窗口紧挨着的下一个时间序列数据点（3 小时）的预测值。因而总共得到 5820 个训练样本（时间窗口），每个时间窗口样本都与上一个时间窗口差值为 3 小时。

将上述 5820 个训练样本的输入值（5820，8）首先通过标准化后，进行 RBF 核主成分

分析升维，降低复杂度，得到（5820，5810）的输入矩阵。随后将其变形为（5820，1，5810）的输入矩阵，再将其输入进 GRU 模型。

本文使用的 GRU 模型网络结构如图4.8所示：

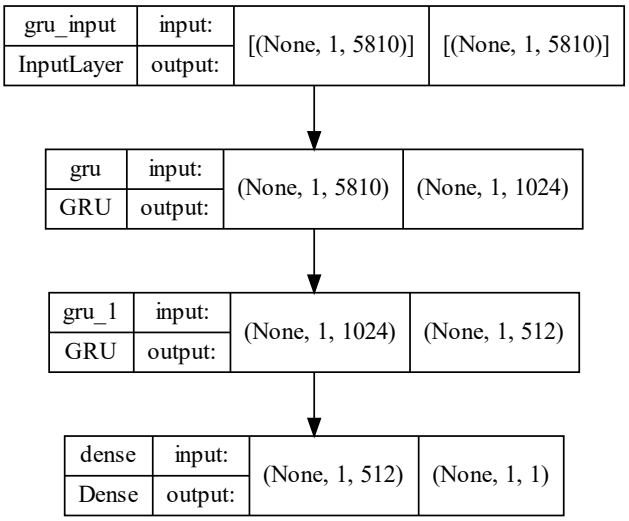


图 4.8　本文使用的 GRU 模型网络结构图

该模型四层之间使用序贯模型的形式进行连接。第一层为模型的输入层，每次输入一个（1，5810）的向量，代表一个样本的时间窗口。第二层为一个使用标准 GRU 模型（即激活函数为 sigmoid，循环激活函数为 tanh）的隐藏层，其一共有 1024 个 GRU 单元，输出为（1，1024）的向量，可训练参数共计 21000192 个。第三层为一个将激活函数和循环激活函数都改为 gelu 的 GRU 模型，其共计 512 个 GRU 单元，输出为（1，512）的向量，可训练参数为 2362368 个。第四层为输出层，将前一层的输出值通过全连接的方式，得到（1，1）的向量，可训练参数 513 个。整个模型可训练参数共计 23363073 个，输出最终预测值。

进行预测时使用多步预测的方法。每个时间窗口的输入为现在已知的数据点值，混合上一个时间窗口的预测输出值，从而进行四步预测（12 小时）。代码部分见附录部分 A.9。

4.4　基于风速历史时间序列数据的 ICEEDMAN-Prophet-GRU 模型

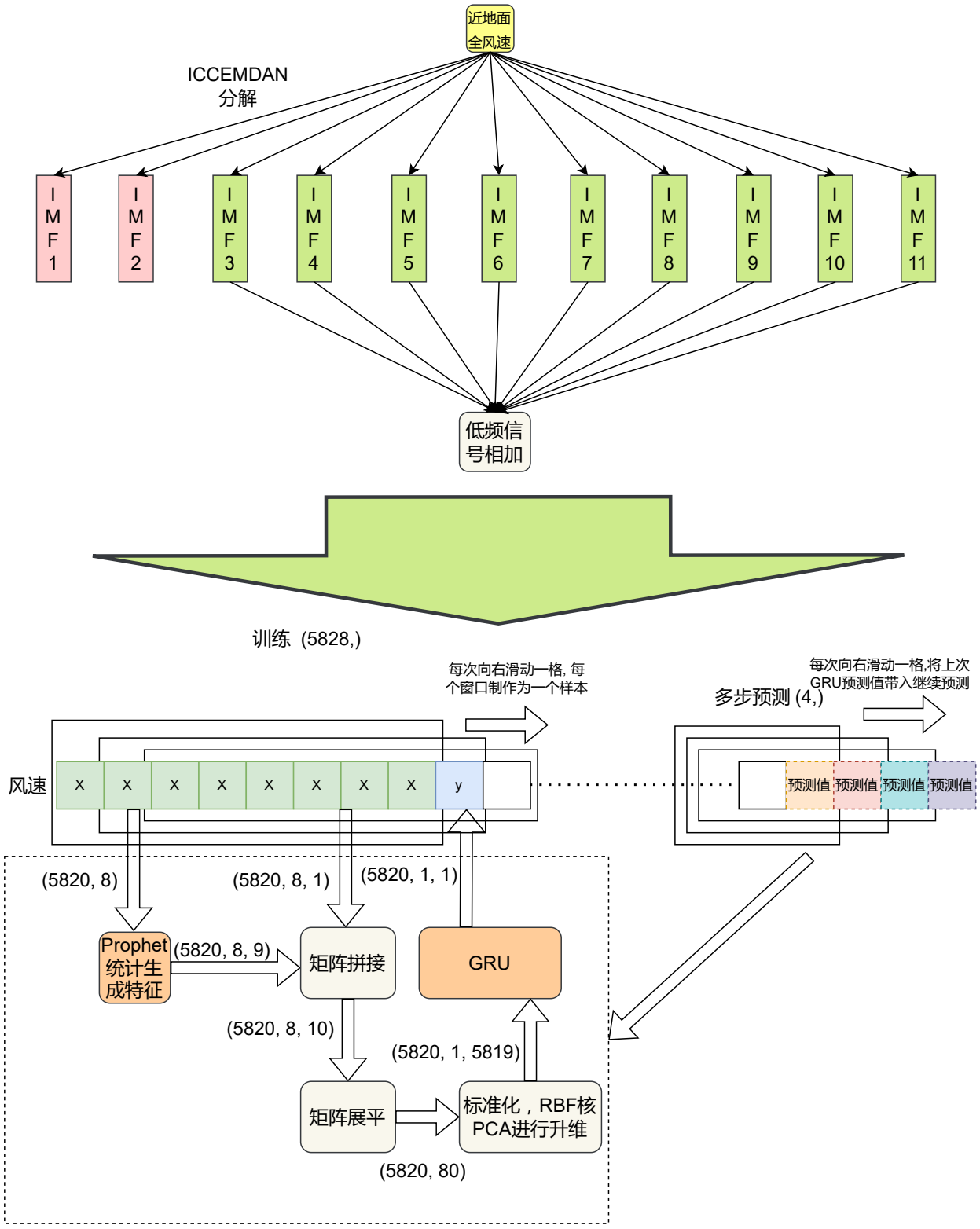


图 4.9　基于风速历史时间序列数据的 ICEEDMAN-Prophet-GRU 模型结构图

如图4.9所示，本模型在基于风速历史时间序列数据的 ICEEDMAN-GRU 模型基础上，将每个窗口的输入值同时使用 Prophet 框架进行统计分析处理，得到该窗口数据对应的趋

势分量、累加式季节性分量以及上述分量所对应的预测最大边界和最小边界共计 9 个特征，并与原始数据进行拼接整合，得到（5820，8，10）的矩阵。随后将后两个维度展平，生成的（5820，80）矩阵进行数据标准化缩放，再通过使用 RBF 核函数的核主成分分析（KPCA），将训练集升维至（5820，5819），变形之后再输入进入同基于风速历史时间序列数据的 ICEEDMAN-GRU 模型中一样的 GRU 模型，从而得到最终的预测结果。代码部分见附录部分 A.10。

4.5　多特征 ICEEDMAN-Prophet-GRU-NN 模型

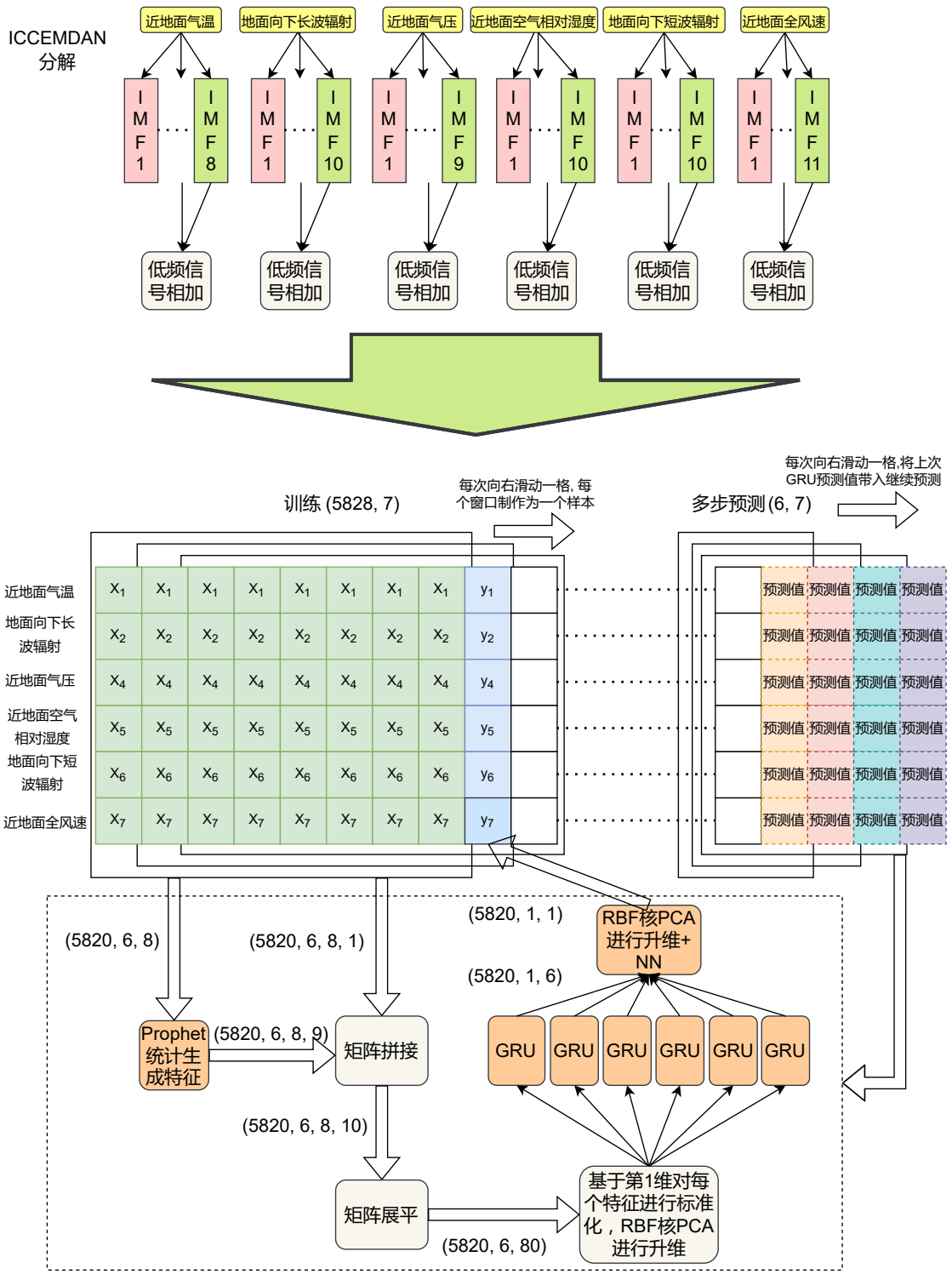


图 4.10　多特征 ICEEDMAN-Prophet-GRU 模型结构图

图4.9展示了多特征 ICEEDMAN-Prophet-GRU 模型结构。该模型将地面向下长波辐射、近地面气压、近地面空气相对湿度、地面向下短波辐射、近地面气温，和近地面全风速 6 种气象要素的原始历史数据的每个气象要素通过和对风速历史时间序列数据的 ICEEDMAN-Prophet-GRU 模型一致的处理方式,得到每个气象要素储窗口样本的 GRU 模型预测值（5820，1，6），再将这些预测值通过使用 RBF 核函数的核主成分分析（KPCA），将其升维至（5820，1，4260），随后使用如图4.11所示的 NN 模型将这些预测值输入，得到最终的对风速的预测结果。

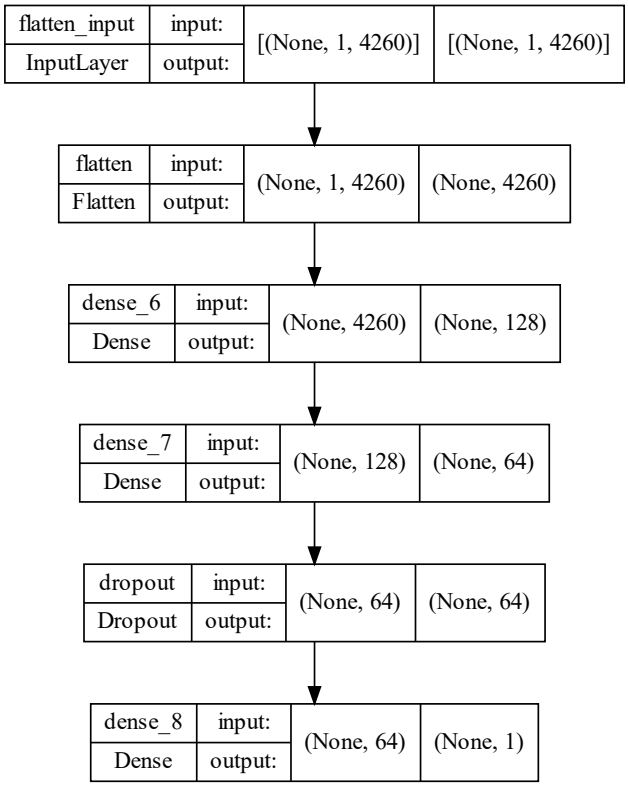


图 4.11　多特征 ICEEDMAN-Prophet-GRU 模型中使用的 NN 模型网络结构图

该神经网络模型（NN）六层之间使用序贯模型的形式进行连接。第一层为模型的输入层，每次输入一个（1，4260）的向量。第二层为一个展平（Flatten）层，将输入展平为（4260）。第三、四层均为全连接层，其激活函数均为 gelu，各有 128、64 个神经元，输出为（64）的向量，可训练参数分别为 555776 个和 8256 个. 第五层为丢弃层（DropOut 层），其丢弃比率为 20%，用于防止神经网络的过拟合。最后一层为输出层，将前一层的输出值通过全连接的方式，得到最后结果，可训练参数 65 个。整个模型可训练参数共计 564097 个，输出最终预测值。代码部分见附录部分 A.11。

4.6 模型评价指标公式

下述指标公式使用 scikit-learn 自带的方法实现，并和 25% 准确率（预测值和真实值实际偏差数值在 25% 以内的比值）封装为 print_metrics 函数调用，相关代码见附录部分 A.6。

1. 均方误差（MSE）

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t) - y(t))^2$$

2. 平均绝对误差（MAE）

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|\hat{y}(t) - y(t)|$$

3. 平均绝对误差百分比（MAPE）

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{\hat{y}(t) - y(t)}{y(t)}\right|$$

4. 均方根误差（RMSE）

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(\hat{y}(t) - y(t))^2}$$

上述四种误差指标均为，当对应指标值越小时，模型的精度越高。

5. 决定系数（R^2）

$$R^2 = 1 - \frac{\sum_{t=1}^{N}(\hat{y}(t) - y(t))^2}{\sum_{t=1}^{N}(\bar{y}(t) - y(t))^2}$$

决定系数（R^2）取值范围为 $(-\infty, 1]$，越接近 1 代表模型的准确度越好。

4.7 模型的对比评估

表 4.2 五种模型在测试集中四步预测对比评估

模型名称	MAPE	MAE	MSE	RMSE	R^2	15% 准确度
基于风速的 Prophet 模型	0.7599	1.5475	2.4874	1.5771	-14.4053	0
基于风速的 ICCEMDAN-Prophet 模型	0.7396	1.5054	2.3588	1.5359	-13.6092	0
基于风速的 ICCEMDAN-GRU 模型	0.1671	0.3	0.2055	0.4533	-0.2728	0.75
风速 ICCEMDAN-Prophet-GRU 模型	0.1356	0.286	0.1121	0.3349	0.3051	0.5
多特征 ICCEMDAN-Prophet-GRU-NN	0.0825	0.1734	0.0307	0.1751	0.81	1

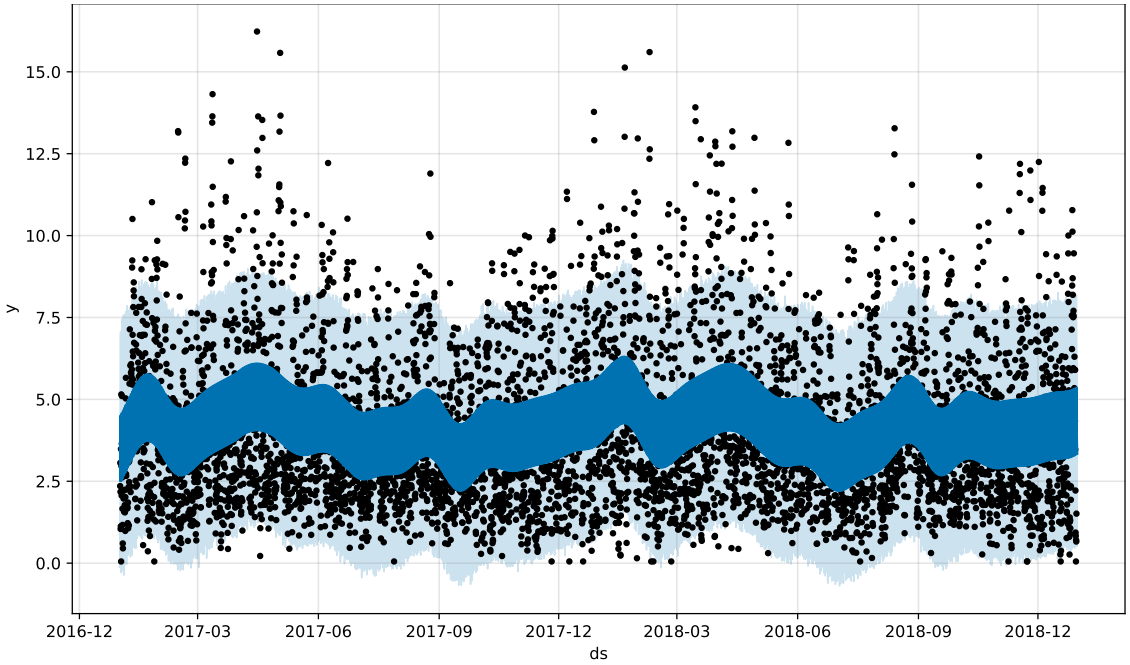
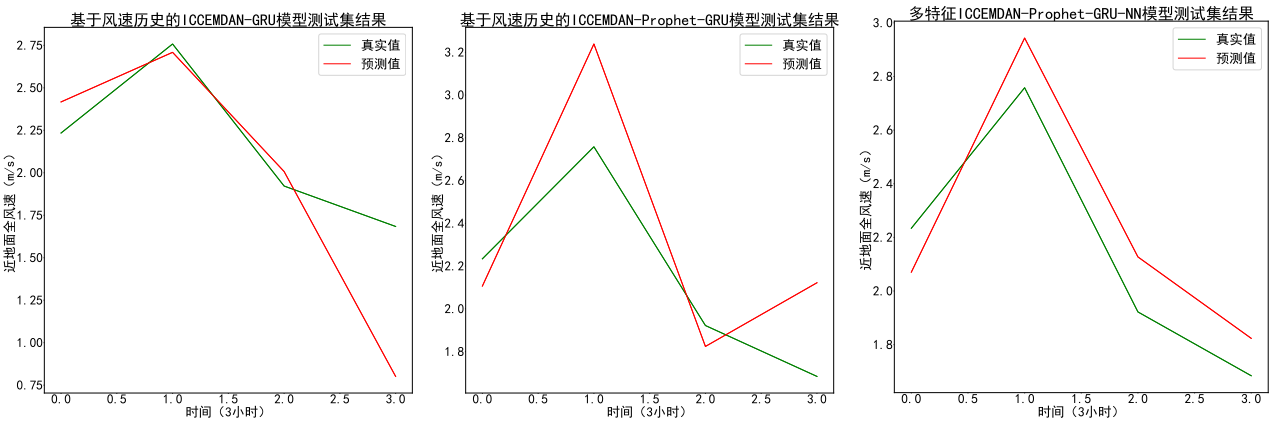


图 4.12　基于风速历史的 Prophet 模型可视化图

图4.12展示了基于风速历史的 Prophet 模型的可视化图，即模型的预测结果与实际值的对比。该模型采用上述章节中搜寻得到的 Prophet 模型最优超参数，分析历史 5828 条数据从而直接预测出未来 12 小时（4 条）的风速值。代码请见附录部分 A.7 和 A.8。图中黑点部分为实际值，实蓝色曲线部分为预测值，透明蓝色部分的上下边界分别为预测结果可能的最大值与最小值。从图中我们可以看到其实际值与预测值的偏差很大，结合该图与表4.2中对未来 4 步预测数据评价指标值可知，Prophet 模型的预测结果很差。这是因为 Prophet 模型类似于传统的 SARIMA 模型，对具有线性关系的时间序列数据预测精度较高，但是对于具有复杂非线性关系的时间序列其很难预测准确。另外，从表4.2中我们还可以看到，对风速数据进行 ICCEMDAN 分解处理，剔除高频噪声后，降低了时间序列数据的复杂度，基于风速历史 ICCEMDAN-Prophet 模型的准确度相较于未经分解的基于风速历史 Prophet 模型而言各项指标都有比较好的提升效果。

神经网络部分的实现使用 Tensorflow Keras。对于所有 GRU 模型，为统一标准，选择训练 10 个 epoch，对于多特征 ICCEMDAN-Prophet-GRU-NN 中 NN 模型选择训练 5000 个 epoch。批大小（Batch Size）为默认 32，训练过程中设置回调函数，当损失 5 步内不再下降时自动停止训练过程，并自动保存训练过程中得到的损失最小的模型。损失函数为 Huber，模型优化器为 Nadam，设定学习率为 0.001。

最终模型用于预测未来四步风速的可视化情况见图4.13，最终模型四步预测的评价指标数值见表4.2。



(a) 风速 ICCEMDAN-GRU　(b) 风速 ICCEMDAN-Prophet-GRU　(c) 多特征 ICCEMDAN-Prophet-GRU-NN

图 4.13　三种本文提出的基于神经网络的模型四步预测风速的可视化图

由图4.13和表4.2可知，使用 GRU 模型之后，模型的各项指标相较于只使用 Prophet 而言都得到了大幅度地显著提高。进一步地，加入 Prophet 的统计分析数据之后交由 GRU 进行训练，模型的准确度和各项指标进一步提升，但是在进行第四步预测时都出现了预测值的较大偏离问题。最终，使用多气象要素特征训练 GRU 模型，并将这些气象要素模型的预测结果输入进 NN，从而对风速的预测值进行校正，产生的预测结果 MAPE 值达到了个位数，准确率有了比较大的提升，且第四步预测时准确度也较好，体现了多特征 ICCEMDAN-Prophet-GRU-NN 模型的优越性和实用价值。

第五章　总结与展望

本文综合使用目前最新的科研成果，基于 ICEEMDAN 特征分解进行了多气象要素的联合预测，而不仅仅是使用风速的历史时间序列。并且采用 Prophet 与 GRU 的组合模型，对相关模型进行优化，突破模型的刻板传统，使用了 RBF 核 PCA 升维、GELU 以及 Nadam 优化器和 Huber 损失函数，并将这些应用到了短期风速预测领域，并通过选择甘肃酒泉的一个真实风力发电厂附近进行短期风速的预测，最终预测结果显示了该模型的优势和极大的实用价值。

当然，因为论文时间紧迫，展望未来，本次论文写作过程也存在着许多遗憾的点：

1. 本次建模过程只选择了两年跨度的时间序列数据，没有将更大尺度的数据进行处理，对于大尺度数据而言模型应该能进一步取得更好的效果。
2. 由于 Prophet 搜寻最优超参数所耗费时间过长，本文只对风速原始时间序列数据这一个模型进行了搜索最优超参数的过程，因为对于一个模型的超参数搜寻时间在作者电脑中运行完成大概就要花上三、四天的时间，对于其他特征时间序列本文则直接沿用了风速原始时间序列数据的最优超参数。如果对每个特征的时间序列 Prophet 模型都加以优化，预计会产生更好的效果。
3. 本文只选择了甘肃酒泉的一个真实风力发电厂附近的地点进行短期风速的预测，当然本文作者坚信在其他地点本模型也会产生很好的效果，后续有待验证这一点。

参考文献

[1] 王庆一. 2020 能源数据 [Z], 2021. https://www.efchina.org/Attachments/Report/report-lceg-20210430-3/2020%E8%83%BD%E6%BA%90%E6%95%B0%E6%8D%AE.pdf.

[2] 李仲蔚. 风力发电企业价值评估研究 [D]. 湖南大学, 2019.

[3] 陆冰鉴, 周鹏, 王兴, et al. 基于 eemd 和 lstm 的短期风速预测模型研究 [J]. 软件工程, 2020, (3):43--48.

[4] 姜兆宇, 贾庆山, 管晓宏. 多时空尺度的风力发电预测方法综述 [J]. 自动化学报, 2019, 45(001):51--71.

[5] 朱智慧, 黄宁立. T639 数值预报产品在南汇站风速预报中的统计释用 [C]. 2010.

[6] 夏晓玲, 尚媛媛, 郑奕. 贵州省数值预报风速产品检验及订正 [J]. 中低纬山地气象, 2019, 43(6):7.

[7] Sun R N, Wang F, University J N. Forecast of wind speed and wing power based on time series[J]. *Computer Knowledge and Technology*, 2017.

[8] Haddad M, Nicod J, Mainassara Y B, et al. Wind and solar forecasting for renewable energy system using sarima-based model[C]. 2019.

[9] Xl A, Zi L B, Zf C. Short-term offshore wind speed forecast by seasonal arima - a comparison against gru and lstm[J]. *Energy*, 2021, 227.

[10] 朱霄旬, 徐搏超, 焦宏超, et al. 遗传算法对 svr 风速预测模型的多参数优化 [J]. 电机与控制学报, 2017, 21(2):6.

[11] Pan C, Tan Q, Cai G, et al. Hybrid short-term wind speed prediction model by coa-svr based on recursive quantitative analysis[J]. *Power System Technology*, 2018.

[12] Shivani, Sandhu K S, Nair A R. A comparative study of arima and rnn for short term wind speed forecasting[C]. 2019.

[13] Alencar D B, Affonso C M, Oliveira R C, et al. Hybrid approach combining sarima and neural networks for multi-step ahead wind speed forecasting in brazil[J]. *IEEE Access*, 2018, 6:55986--55994.

[14] 朱丽娜. 风电场短期风速预测方法研究 [D]. 兰州理工大学, 2021.

[15] 谢义超. 基于 ceemdan 分解和改进的 lstm 模型的短期风速预测 [D]. 武汉科技大学, 2021.

[16] 王秀. 基于 wd-arima-lstm 的短期风速预测 [D]. 华北电力大学, 2021.

[17] 阳坤, 何杰. 中国区域地面气象要素驱动数据集（1979-2018）[J]. 国家青藏高原科学数据中心, 2019.

[18] On downward shortwave and longwave radiations over high altitude regions: Observation and modeling in the tibetan plateau[J]. 2010.

[19] The first high-resolution meteorological forcing dataset for land process studies over china[J].

[20] Huang N E, Shen Z, Long S R, et al. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis[J]. *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences*, 1998, 454(1971):903--995.

[21] Wu Z, Huang N E. Ensemble empirical mode decomposition: a noise-assisted data analysis method[J]. *Advances in adaptive data analysis*, 2009, 1(01):1--41.

[22] Torres M E, Colominas M A, Schlotthauer G, et al. A complete ensemble empirical mode decomposition with adaptive noise[C]. 2011. 4144--4147.

[23] Colominas M A, Schlotthauer G, Torres M E. Improved complete ensemble emd: A suitable tool for biomedical signal processing[J]. *Biomedical Signal Processing and Control*, 2014, 14:19--29.

[24] Taylor S J, Letham B. Forecasting at scale[J]. *The American Statistician*, 2018, 72(1):37--45.

[25] Jolliffe I T, Cadima J. Principal component analysis: a review and recent developments[J]. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2016, 374(2065):20150202.

[26] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. *nature*, 1986, 323(6088):533--536.

[27] Hochreiter S, Schmidhuber J. Long short-term memory[J]. *Neural computation*, 1997, 9(8):1735--1780.

[28] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation[J]. *arXiv preprint arXiv:1406.1078*, 2014.

[29] Chung J, Gulcehre C, Cho K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. *arXiv preprint arXiv:1412.3555*, 2014.

[30] Hendrycks D, Gimpel K. Gaussian error linear units (gelus)[J]. *arXiv preprint arXiv:1606.08415*, 2016.

[31] Ruder S. An overview of gradient descent optimization algorithms[J]. *arXiv preprint arXiv:1609.04747*, 2016.

[32] Qian N. On the momentum term in gradient descent learning algorithms[J]. *Neural networks*, 1999, 12(1):145--151.

[33] Nesterov Y. A method for unconstrained convex minimization problem with the rate of convergence o (1/k^ 2)[C]. 1983. 543--547.

[34] Bengio Y, Boulanger-Lewandowski N, Pascanu R. Advances in optimizing recurrent networks[C]. 2013. 8624--8628.

[35] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. *arXiv preprint arXiv:1412.6980*, 2014.

[36] Dozat T. Incorporating nesterov momentum into adam[J]. 2016.

[37] Huber P J. Robust estimation of a location parameter[C]. Proceedings of Breakthroughs in statistics. Springer, 1992: 492--518.

附　　录

A.1　数据集制作代码

```
from netCDF4 import Dataset
import datetime

# 甘肃中电酒泉第四风力发电有限公司
lat_index = 256
lon_index = 269
variable_list = ["lrad", "prec", "pres", "shum",
                 "srad", "temp", "wind"]
with open('data.csv', 'w') as f:
    f.write("ds")
    for variable in variable_list:
        f.write(',' + variable)
    f.write('\n')
    for year in range(2017, 2019):
        for month in range(1, 13):
            dataset = []
            time_data = []
            for name in variable_list:
                data = Dataset("data/" + name +
                               "_ITPCAS-CMFD_V0106_B-01_03hr_010deg_"
                               + str(year) + str(month).zfill(2) +
                               ".nc")
                dataset.append(data.variables[name][:])
                time_data = data.variables['time'][:]

            for index in range(len(time_data)):
                f.write(datetime.datetime.utcfromtimestamp(
                    (time_data[index]-613608)*3600).strftime(
                        "%Y-%m-%d %H:%M:%S"))
                for data in dataset:
                    f.write(',' + str(
                            data[index][lat_index][lon_index]))
                f.write('\n')
```

A.2 表2.2 中的数据源制作代码

```
import pandas as pd
data = pd.read_csv('data.csv')
stat = data.describe()
stat.loc['range'] = stat.loc['max']-stat.loc['min']
stat.loc['dis'] = stat.loc['75%']-stat.loc['25%']
stat.loc['var'] = stat.loc['std']/stat.loc['mean']
stat.loc['mad'] = data.mad()
stat.loc['skew'] = data.skew()
stat.loc['kurt'] = data.kurt()
print(stat)
```

A.3 表2.3 中的数据源制作代码

```
from scipy import stats
for item in ['lrad', 'prec', 'pres', 'shum', 'srad', 'temp']:
    print(stats.spearmanr(
          data[[item]].to_numpy().ravel(),
          data[['wind']].to_numpy().ravel()))
```

A.4 图4.1-图4.6 制作代码以及 ICEEMDAN 分解数据的存储

```
import matplotlib.pyplot as plt
plt.rcParams['font.size']=64
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

import numpy as np
from PyEMD import CEEMDAN
def draw_spectrum_map(x):
    xf = np.fft.fft(x)
    xf_abs = np.fft.fftshift(abs(xf))[len(x)//2:]
    plt.plot(xf_abs)

IImfs=[]
def ceemdan_decompose_res(data, trials=100, epsilon=0.005, noise_scale=1,
    noise_kind="normal", range_thr=0.01, total_power_thr=0.05):
    ceemdan = CEEMDAN(parallel=True, processes=8, trials=trials, epsilon=
        epsilon, noise_scale=noise_scale, noise_kind=noise_kind, range_thr=
```

```
        range_thr, total_power_thr=total_power_thr)
    ceemdan.ceemdan(data)
    imfs, res = ceemdan.get_imfs_and_residue()
    plt.subplots_adjust(hspace=0.5)
    plt.subplot(imfs.shape[0]+3, 2, 1)
    plt.plot(data,'r')
    plt.title(u"时域")
    plt.ylabel(u"原始数据")
    plt.subplot(imfs.shape[0]+3, 2, 2)
    draw_spectrum_map(data)
    plt.title(u"频域")
    for i in range(imfs.shape[0]):
        plt.subplot(imfs.shape[0]+3, 2, 2*i+3)
        plt.plot(imfs[i], 'g')
        plt.ylabel("IMF %i" %(i+1))
        plt.locator_params(axis='x', nbins=10)
        plt.subplot(imfs.shape[0]+3, 2, 2*i+4)
        draw_spectrum_map(imfs[i])
        IImfs.append(imfs[i])
    plt.subplot(imfs.shape[0]+3, 2, 2*imfs.shape[0]+3)
    plt.plot(res,'g')
    plt.ylabel(u"残差")
    plt.subplot(imfs.shape[0]+3, 2, 2*imfs.shape[0]+4)
    draw_spectrum_map(res)
    return res
count = 0
for name in ["lrad", "prec", "pres", "shum", "srad", "temp", "wind"]:
    pic = plt.figure(figsize=(64,64))
    input_data = data[name].to_numpy().ravel()
    ceemdan_decompose_res(input_data)
    with open(name + '.csv', 'w') as f:
        f.write("ds")
        for i in range(1, len(IImfs)-count+1):
            f.write(',IMF' + str(i))
        f.write('\n')
        for n in range(len(input_data)):
            f.write(data['ds'][n])
            for index in range(count, len(IImfs)):
                f.write(',' + str(IImfs[index][n]))
            f.write('\n')
```

```
    count=len(IImfs)
    pic.savefig(name + ".pdf")

imf_dic={'lrad':2, 'pres':2, 'shum':2, 'srad':2, 'temp':2, 'wind':2}
data_decomposed = pd.DataFrame()
data_decomposed['ds'] = data['ds']
for feature in imf_dic.keys():
    data_iceemdan = pd.read_csv(feature+'.csv')
    temp = pd.DataFrame({feature:[0.0 for _ in range(len(data_iceemdan))
        ]})
    for index in data_iceemdan.columns:
        if index.startswith('IMF') and imf_dic[feature] < int(index.lstrip
            ('IMF')):
            temp[feature] += data_iceemdan[index]
    data_decomposed=data_decomposed.join(temp)
data_decomposed.to_csv('data_decomposed.csv')
data_decomposed
```

A.5 基于风速历史时间序列数据的 Prophet 模型的参数交叉验证调优

```
import itertools
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
param_grid = {
    'changepoint_prior_scale': [0.001, 0.01, 0.1, 0.5, 1.0, 10.0, 100.0,
        1000.0, 10000.0],
    'seasonality_prior_scale': [0.01, 0.1, 1.0, 10.0, 100.0, 1000.0,
        10000.0],
    'seasonality_mode': ['additive', 'multiplicative'],
    'changepoint_range': [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
}

all_params = [dict(zip(param_grid.keys(), v)) for v in itertools.product(*
    param_grid.values())]
mapes = []

df = pd.DataFrame(data['ds'])
df['y'] = data['wind']

```

```
for params in all_params:
    m = Prophet(yearly_seasonality=True, **params).fit(df)
    df_cv = cross_validation(m, initial='438 days', period='5 days',
        horizon = '12H', parallel="processes")
    df_p = performance_metrics(df_cv, rolling_window=1)
    mapes.append(df_p['mape'].values[0])

tuning_results = pd.DataFrame(all_params)
tuning_results['mape'] = mapes

best_params = all_params[np.argmin(mapes)]
print(best_params)
```

A.6 模型评价指标实现代码

```
from sklearn import metrics
def print_metrics(pred, y_vals):
    print('mape: ', metrics.mean_absolute_percentage_error(y_vals, pred))
    print('mae: ', metrics.mean_absolute_error(y_vals, pred))
    print('mse: ', metrics.mean_squared_error(y_vals, pred))
    print('rmse: ', np.sqrt(metrics.mean_squared_error(y_vals, pred)))
    print('r2: ', metrics.r2_score(y_vals, pred))
    count = 0
    y_error = pred.flatten() - y_vals.flatten()
    y_error = np.array([abs(e) for e in y_error]).flatten()
    for i in range(len(y_error)):
        if(y_error[i] < 0.15 * y_vals[i]):
            count += 1
    print('15% 准确度: ', count / len(pred))
```

A.7 基于风速历史时间序列数据的 Prophet 模型实现代码和相关指标计算、可视化对比图形的制作

```
from prophet import Prophet
split_line=5828
df = pd.DataFrame(data['ds'][:split_line])
df['y'] = data['wind'][:split_line]
m = Prophet(changepoint_prior_scale=1.0, seasonality_prior_scale=0.1,
    seasonality_mode='additive', changepoint_range=1, daily_seasonality=
```

```
    True, yearly_seasonality=True, weekly_seasonality=False)
m.fit(df)
future = m.make_future_dataframe(periods=4, freq='3H')
forecast = m.predict(future)
fig = m.plot(forecast)
fig.savefig("prophet_wind.pdf")

print_metrics(forecast['yhat'][split_line:].to_numpy(), data['wind'][
    split_line:].to_numpy())
```

A.8 基于风速历史时间序列数据的 ICEEDMAN-Prophet 模型实现代码和相关指标计算、可视化对比图形的制作

```
data = pd.read_csv('data_decomposed.csv')
df = pd.DataFrame(data['ds'][:split_line])
df['y'] = data['wind'][:split_line]
m = Prophet(changepoint_prior_scale=1.0, seasonality_prior_scale=0.1,
    seasonality_mode='additive', changepoint_range=1, daily_seasonality=
    True, yearly_seasonality=True, weekly_seasonality=False)
m.fit(df)
future = m.make_future_dataframe(periods=4, freq='3H')
forecast = m.predict(future)
fig = m.plot(forecast)
fig.savefig("prophet_wind_decomposed.pdf")

print_metrics(forecast['yhat'][split_line:].to_numpy(), data['wind'][
    split_line:].to_numpy())
```

A.9 基于风速历史时间序列数据的 ICEEDMAN-GRU 模型实现代码和相关指标计算、可视化对比图形的制作

```
feature="wind"
window = 8
predict_num = 1
x_row_list = []
y_row_list = []
for row in range(split_line-window-predict_num+1):
    x_row_list.append(dict(("x"+str(index),data[feature][row+index]) for
        index in range(window)))
```

```
    y_row_list.append(dict(("y"+str(index),data[feature][row+index+window
        ]) for index in range(predict_num)))
X=pd.DataFrame(x_row_list)
y=pd.DataFrame(y_row_list)

from sklearn.preprocessing import StandardScaler
yscaler = StandardScaler()
y_train = yscaler.fit_transform(y.to_numpy())
scaler = StandardScaler()
X_train = scaler.fit_transform(X.to_numpy())
from sklearn.decomposition import KernelPCA
pca = KernelPCA(kernel='rbf')
X_train = pca.fit_transform(X_train)
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))

from tensorflow import keras
modelname='wind_gru'
if os.path.isdir(modelname):
    model = keras.models.load_model(modelname)
else:
    model = keras.models.Sequential()
    model.add(keras.layers.GRU(1024,input_shape = (predict_num, X_train.
        shape[2]),return_sequences = True))
    model.add(keras.layers.GRU(512,activation = 'gelu',
        recurrent_activation = 'gelu',return_sequences = True))
    model.add(keras.layers.Dense(predict_num))
    es_callback = keras.callbacks.EarlyStopping(patience = 1,
        restore_best_weights = True, monitor="loss")
    model.compile(loss = keras.losses.Huber(), optimizer = keras.
        optimizers.Nadam(0.001))
    model.summary()
    keras.utils.plot_model(model, to_file=modelname+'model_plot.pdf',
        show_shapes=True, show_layer_names=True)
    history=model.fit(X_train, y_train, epochs = 10, verbose = 1, shuffle
        = True, callbacks = [es_callback])
    model.save(modelname)

predicted = yscaler.inverse_transform(model.predict(X_train)).flatten()
predicted[predicted < 0] = 0
row_list = y_row_list.copy()
```

```
for pointer in range(X.shape[0],len(data)-split_line+X.shape[0]):
    input_np = np.array([row_list[row]['y0'] for row in range(pointer-
        window, pointer)])
    input_np = scaler.transform(input_np.reshape(1, -1))
    input_np = pca.transform(input_np)
    input_np = np.reshape(input_np, (input_np.shape[0], 1, input_np.shape
        [1]))
    row_list.append(dict(("y"+str(index),yscaler.inverse_transform(model.
        predict(input_np))[0][index][0]) for index in range(predict_num)))

predictions = pd.DataFrame(row_list)[X.shape[0]:]
predictions[predictions < 0] = 0

y_train = y.to_numpy()
print_metrics(predictions.to_numpy(), data[feature][split_line:].to_numpy
    ())
generated = pd.DataFrame(data['ds'])[window:]
generated['y_pred'] = np.append(predicted, predictions)
generated['y_real'] = np.array(pd.DataFrame(data[feature])[window:])
generated.to_csv(modelname + ".csv")

import matplotlib.pyplot as plt
plt.rcParams['font.size']=64
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

pic = plt.figure(figsize=(32,32))
plt.plot(data[feature][split_line:].to_numpy(),'g',predictions.to_numpy(),
    'r')
plt.title('基于风速历史的ICEEDMAN-GRU模型测试集结果')
plt.plot(data[feature][split_line:].to_numpy(),'g',label=u'真实值')
plt.plot(predictions.to_numpy(),'r',label=u'预测值')
plt.xlabel(u'时间（3小时）')
plt.ylabel(u'近地面全风速（m/s）')
plt.legend()
pic.savefig(modelname + "_predict_test.pdf")
```

A.10 基于风速历史时间序列数据的 ICEEDMAN-Prophet-GRU 模型实现代码和相关指标计算、可视化对比图形的制作

```
filename='only_prophet_gru_wind.npy'
features = ['trend', 'yhat_lower', 'yhat_upper', 'trend_lower', '
    trend_upper', 'additive_terms',
             'additive_terms_lower', 'additive_terms_upper', 'yhat']
if os.path.isfile(filename):
    X=np.load(filename)
else:
    X=np.empty((0,(len(features)+1)*window))
    for row in range(split_line-window-predict_num+1):
        temp_df = pd.DataFrame(data['ds'][row:row+window])
        temp_df['y'] = data['wind'][row:row+window]
        m = Prophet(changepoint_prior_scale=1.0, seasonality_prior_scale
            =0.1, seasonality_mode='additive', changepoint_range=1,
            n_changepoints=window-1)
        m.fit(temp_df)
        future = m.make_future_dataframe(periods=1, freq='3H')
        forecast = m.predict(future)
        forecast = forecast[features].to_numpy()[:-1]
        forecast = np.append(forecast, np.array([data['wind'][row+i] for i
            in range(window)]).reshape(-1,1), axis=1)
        forecast = forecast.flatten()
        forecast = forecast.reshape(1,-1)
        X=np.append(X, forecast, axis=0)
    np.save(filename, X)
scaler = StandardScaler()
X_train = scaler.fit_transform(X)
from sklearn.decomposition import KernelPCA
pca = KernelPCA(kernel='rbf')
X_train = pca.fit_transform(X_train)
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))

from tensorflow import keras
modelname='wind_only_prophet_gru'
if os.path.isdir(modelname):
    model = keras.models.load_model(modelname)
else:
    model = keras.models.Sequential()
    model.add(keras.layers.GRU(1024,input_shape = (predict_num, X_train.
        shape[2]),return_sequences = True))
```

```
35      model.add(keras.layers.GRU(512,activation = 'gelu',
            recurrent_activation = 'gelu',return_sequences = True))
36      model.add(keras.layers.Dense(predict_num))
37      es_callback = keras.callbacks.EarlyStopping(patience = 5,
            restore_best_weights = True, monitor="loss")
38      model.compile(loss = keras.losses.Huber(), optimizer = keras.
            optimizers.Nadam(0.001))
39      model.summary()
40      keras.utils.plot_model(model, to_file=modelname+'model_plot.pdf',
            show_shapes=True, show_layer_names=True)
41      history=model.fit(X_train, y_train, epochs = 10, verbose = 1, shuffle
            = True, callbacks = [es_callback])
42      model.save(modelname)
43
44  predicted = yscaler.inverse_transform(model.predict(X_train)).flatten()
45  predicted[predicted < 0] = 0
46  row_list = y_row_list.copy()
47  for pointer in range(X.shape[0],len(data)-split_line+X.shape[0]):
48      input_np = np.array([row_list[row]['y0'] for row in range(pointer-
            window, pointer)])
49      temp_df = pd.DataFrame(data['ds'][pointer:pointer+window])
50      temp_df['y'] = input_np
51      m = Prophet(changepoint_prior_scale=1.0, seasonality_prior_scale=0.1,
            seasonality_mode='additive', changepoint_range=1, n_changepoints=
            window-1)
52      m.fit(temp_df)
53      future = m.make_future_dataframe(periods=1, freq='3H')
54      forecast = m.predict(future)
55      forecast = forecast[features].to_numpy()[:-1]
56      forecast = np.append(forecast, np.array([row_list[row]['y0'] for row
            in range(pointer-window, pointer)]).reshape(-1,1), axis=1)
57      input_np = forecast.flatten()
58      input_np = scaler.transform(input_np.reshape(1, -1))
59      input_np = pca.transform(input_np)
60      input_np = np.reshape(input_np, (input_np.shape[0], 1, input_np.shape
            [1]))
61      row_list.append(dict(("y"+str(index),yscaler.inverse_transform(model.
            predict(input_np))[0][index][0]) for index in range(predict_num)))
62
63  predictions = pd.DataFrame(row_list)[X.shape[0]:]
```

```
predictions[predictions < 0] = 0

y_train = y.to_numpy()
print_metrics(predictions.to_numpy(), data["wind"][split_line:].to_numpy()
    )

generated = pd.DataFrame(data['ds'])[window:]
generated['y_pred'] = np.append(predicted, predictions)
generated['y_real'] = np.array(pd.DataFrame(data['wind'])[window:])
generated.to_csv(modelname + ".csv")

import matplotlib.pyplot as plt
plt.rcParams['font.size']=64
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

pic = plt.figure(figsize=(32,32))
plt.plot(data["wind"][split_line:].to_numpy(),'g',predictions.to_numpy(),'
    r')
plt.title('基于风速历史的ICCEMDAN-Prophet-GRU模型测试集结果')
plt.plot(data["wind"][split_line:].to_numpy(),'g',label=u'真实值')
plt.plot(predictions.to_numpy(),'r',label=u'预测值')
plt.xlabel(u'时间（3小时）')
plt.ylabel(u'近地面全风速（m/s）')
plt.legend()
pic.savefig(modelname + "_predict_test.pdf")
```

A.11 多特征 ICEEDMAN-Prophet-GRU-NN 模型实现代码实现代码和相关指标计算、可视化对比图形的制作

```
y_row_list = []
weather_features = ["lrad", "pres", "shum", "srad", "temp", "wind"]
for row in range(split_line-window):
    y_row_list.append(dict(("y"+weather_feature,data[weather_feature][row+
        window]) for weather_feature in weather_features))
y=pd.DataFrame(y_row_list)
from sklearn.preprocessing import StandardScaler
yscaler = StandardScaler()
y_train = yscaler.fit_transform(y.to_numpy())
```

```
9
10 filename='all_feature_prophet_gru.npy'
11 if os.path.isfile(filename):
12     X=np.load(filename)
13 else:
14     X=np.empty((0,(len(features)+1)*window*len(weather_features)))
15     for row in range(split_line-window):
16         temp_array=np.empty((1,0))
17         for weather_feature in weather_features:
18             temp_df = pd.DataFrame(data['ds'][row:row+window])
19             temp_df['y'] = data[weather_feature][row:row+window]
20             m = Prophet(changepoint_prior_scale=1.0,
                   seasonality_prior_scale=0.1, seasonality_mode='additive',
                   changepoint_range=1, n_changepoints=window-1)
21             m.fit(temp_df)
22             future = m.make_future_dataframe(periods=1, freq='3H')
23             forecast = m.predict(future)
24             forecast = forecast[features].to_numpy()[:-1]
25             forecast = np.append(forecast, np.array([data[weather_feature
                   ][row+i] for i in range(window)]).reshape(-1,1), axis=1)
26             forecast = forecast.flatten()
27             forecast = forecast.reshape(1,-1)
28             temp_array=np.append(temp_array, forecast, axis=1)
29         X=np.append(X, temp_array, axis=0)
30     np.save(filename, X)
31
32 tf.config.set_visible_devices([], 'GPU')
33 modelname='all_feature_prophet_gru'
34 scaler_list=[]
35 pca_list=[]
36 data_list=[]
37 model_list=[]
38 for weather_feature_index in range(len(weather_features)):
39     modelfname=modelname+ '_' + weather_features[weather_feature_index]
40     scaler = StandardScaler()
41     scaler_list.append(scaler)
42     X_train = scaler.fit_transform(X.reshape(split_line-window,len(
           weather_features),window,-1)[:,weather_feature_index].reshape(
           split_line-window,-1))
43     pca = KernelPCA(kernel='rbf')
```

```
44      pca_list.append(pca)
45      X_train = pca.fit_transform(X_train)
46      X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
47      data_list.append(X_train)
48      if os.path.isdir(modelfname):
49          model = keras.models.load_model(modelfname)
50      else:
51          model = keras.models.Sequential()
52          model.add(keras.layers.GRU(1024,input_shape = (1, X_train.shape
                [2]),return_sequences = True))
53          model.add(keras.layers.GRU(512,activation = 'gelu',
                recurrent_activation = 'gelu',return_sequences = True))
54          model.add(keras.layers.Dense(1))
55          es_callback = keras.callbacks.EarlyStopping(patience = 5,
                restore_best_weights = True, monitor="loss")
56          model.compile(loss = keras.losses.Huber(), optimizer = keras.
                optimizers.Nadam(0.001))
57          model.summary()
58          keras.utils.plot_model(model, to_file=modelfname+'model_plot.pdf',
                 show_shapes=True, show_layer_names=True)
59          print(weather_features[weather_feature_index])
60          history=model.fit(X_train, y_train[:,weather_feature_index].
                reshape(-1,1), epochs = 10, verbose = 1, shuffle = True,
                callbacks = [es_callback])
61          model.save(modelfname)
62      model_list.append(model)
63
64  row_list = y_row_list.copy()
65  for pointer in range(X.shape[0],len(data)-split_line+X.shape[0]):
66      gru_output=np.empty((1,1,0))
67      for weather_feature_index in range(len(weather_features)):
68          input_np = np.array([row_list[row]['y'+weather_features[
                weather_feature_index]] for row in range(pointer-window,
                pointer)])
69          temp_df = pd.DataFrame(data['ds'][pointer:pointer+window])
70          temp_df['y'] = input_np
71          m = Prophet(changepoint_prior_scale=1.0, seasonality_prior_scale
                =0.1, seasonality_mode='additive', changepoint_range=1,
                n_changepoints=window-1)
72          m.fit(temp_df)
```

```
            future = m.make_future_dataframe(periods=1, freq='3H')
            forecast = m.predict(future)
            forecast = forecast[features].to_numpy()[:-1]
            forecast = np.append(forecast, np.array([row_list[row]['y'+
                weather_features[weather_feature_index]] for row in range(
                pointer-window, pointer)]).reshape(-1,1), axis=1)
            forecast = forecast.flatten().reshape(1,-1)
            input_np = scaler_list[weather_feature_index].transform(forecast)
            input_np = pca_list[weather_feature_index].transform(input_np)
            input_np = np.reshape(input_np, (input_np.shape[0], 1, input_np.
                shape[1]))
            output_np = model_list[weather_feature_index].predict(input_np)
            gru_output=np.append(gru_output, output_np, axis=2)
        row_list.append(dict(("y"+weather_features[index],yscaler.
            inverse_transform(gru_output)[:,:,index][0][0]) for index in range(
            len(weather_features))))
predictions = yscaler.transform(pd.DataFrame(row_list)[X.shape[0]:].
    to_numpy()).reshape(-1,1,len(weather_features))
y_train = y.to_numpy()

gru_output=np.empty((X.shape[0],1,0))
for index in range(len(data_list)):
    predicted = model_list[index].predict(data_list[index])
    predicted = predicted[:,:,-1].reshape(X.shape[0],1,-1)
    gru_output=np.append(gru_output, predicted, axis=2)

if os.path.isdir(modelname):
    model = keras.models.load_model(modelname)
else:
    model = keras.models.Sequential()
    model.add(keras.layers.Flatten(input_shape=(1, len(weather_features)))
        )
    model.add(keras.layers.Dense(128, activation='gelu'))
    model.add(keras.layers.Dropout(0.2))
    model.add(keras.layers.Dense(1))
    es_callback = keras.callbacks.EarlyStopping(patience = 50,
        restore_best_weights = True, monitor="loss")
    model.compile(loss = keras.losses.Huber(), optimizer = keras.
        optimizers.Nadam(0.001))
    model.summary()
```

```
    keras.utils.plot_model(model, to_file=modelname+'model_plot.pdf',
        show_shapes=True, show_layer_names=True)
    history=model.fit(gru_output, y_train.reshape(-1,1,1), epochs = 5000,
        verbose = 1, shuffle = True, callbacks = [es_callback])
    model.save(modelname)

predicted = model.predict(gru_output).flatten()
predicted[predicted < 0] = 0

predictions = model.predict(predictions).flatten()
predictions[predictions < 0] = 0

print_metrics(predictions, data["wind"][split_line:].to_numpy())

generated = pd.DataFrame(data['ds'])[window:]
generated['y_pred'] = np.append(predicted, predictions)
generated['y_real'] = np.array(pd.DataFrame(data['wind'])[window:])
generated.to_csv(modelname + ".csv")

import matplotlib.pyplot as plt
plt.rcParams['font.size']=64
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

pic = plt.figure(figsize=(32,32))
plt.plot(data["wind"][split_line:].to_numpy(),'g',predictions,'r')
plt.title('多特征ICCEMDAN-Prophet-GRU-NN模型测试集结果')
plt.plot(data["wind"][split_line:].to_numpy(),'g',label=u'真实值')
plt.plot(predictions,'r',label=u'预测值')
plt.xlabel(u'时间（3小时）')
plt.ylabel(u'近地面全风速（m/s）')
plt.legend()
pic.savefig(modelname + "_predict_test.pdf")
```

A.12 文中使用的函数图像绘制代码

```
import matplotlib.pyplot as plt
import numpy as np
right=4
left=-right
```

```
x = np.arange(left, right, 0.01)
def initialize_plot():
    plt.xlim(left,right)
    plt.ylim(left,right)
    ax = plt.gca()
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    ax.xaxis.set_ticks_position('bottom')
    ax.yaxis.set_ticks_position('left')
    ax.spines['bottom'].set_position(('data',0))
    ax.spines['left'].set_position(('data',0))

name='sigmoid'
fig = plt.figure(figsize=(6,6))
initialize_plot()
y = 1.0/(1.0+np.exp(-x))
plt.plot(x, y, label = name)
plt.legend()
fig.savefig(name + ".pdf")

name='tanh'
fig = plt.figure(figsize=(6,6))
initialize_plot()
y = (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))
plt.plot(x, y, label = name)
plt.legend()
fig.savefig(name + ".pdf")

fig = plt.figure(figsize=(6,6))
initialize_plot()
ReLU = np.where(x < 0, 0, x)
GELU = 0.5*x*(1+ np.tanh(np.sqrt(2/np.pi)*(x+0.044715*np.power(x,3))))
plt.plot(x, ReLU, label = 'ReLU')
plt.plot(x, GELU, label = 'GELU')
plt.legend()
fig.savefig("relu_gelu.pdf")

name='mse'
fig = plt.figure(figsize=(6,6))
initialize_plot()
```

```
y = x**2
plt.plot(x, y, label = name)
plt.legend([name])
fig.savefig(name + ".pdf")

name='mae'
fig = plt.figure(figsize=(6,6))
initialize_plot()
y = abs(x)
plt.plot(x, y, label = name)
plt.legend([name])
fig.savefig(name + ".pdf")

def huber_loss(x, d):
    return (abs(x)<=d)*x**2/2 + (abs(x)>d)*d*(abs(x)-d/2)
fig = plt.figure(figsize=(6,6))
initialize_plot()
plt.plot(x, x**2/2, label=r'$x^2/2$')
plt.plot(x, huber_loss(x, 1), label=r'huber: $\delta$=1')
plt.legend()
fig.savefig("huber.pdf")
```

致　谢

时光荏苒，岁月如梭。转眼间，近四年的本科生活就要结束了。这四年是充满了挑战和挫折的四年，同时也是充满了丰收和果实的四年。值此毕业论文致谢之际，我首先需要特别感谢的是我的毕业论文指导老师，兰州大学信息科学与工程学院的任超副教授。任老师治学严谨，对我的论文写作极其认真负责。每当我遇到困难请教任老师时，任老师总能事无巨细地耐心讲解。同时，任老师极高的专业素养与渊博的学识也十分令我钦佩！在任老师带领的论文写作过程中，我的科研素养得到了极大地提高，获得了许多十分宝贵的知识和经验。

其次，我要感谢兰州大学提供的一流教学环境与教育资源，兰州大学信息科学与工程学院的各位老师的专业教导，以及兰州大学信息科学与工程学院 2018 级计算机科学与技术基础理论班的各位同学，以及舍友和其他同学的支持与鼓励。四年的朝夕相处，是他们让我拓宽了视野，掌握了基本的计算机专业知识和技能，并且让我获得了学习的动力，为我的继续深造以及未来工作打好了坚实的基础，从而让我能够更好地为社会贡献自己的力量。在后续的学习工作生活中，我必将以梦为马，不负韶华！

最后，我要感谢大学四年父母对我的默默关爱和支持，让我能够顺利地完成本科学业。父母对我的无私且伟大的爱，是我在黑暗中的灯塔，给予了我不断前行的动力。

另外，本文使用的数据集下载于“国家青藏高原科学数据中心”(http://data.tpdc.ac.cn)。在写作时从 GitHub 中获取并使用了兰州大学 2016 级物理科学与技术学院本科生余航制作的“兰州大学本科生 2021 学士学位毕业论文 LaTeX 模板”，在此也一并致谢。

蒋嵩林

2022 年 5 月于兰州大学榆中校区萃英山下