分 类 号_____　　学号__D201880***__
学校代码__10487__　　密级_____

華中科技大學

博士学位论文

(学术型☑　　专业型□)

基于随机映射的时间序列深度学习预测建模技术

学位申请人：　张心泽

学 科 专 业：　管理科学与工程

指 导 教 师：　鲍玉昆　教授

答 辩 日 期：　2023 年 4 月 26 日

答辩委员会

	姓名	职称	单位
主席	X　X	教授	武汉大学 信息管理学院
委员	X　X	教授	武汉大学 信息管理学院
	XXX	教授	华中科技大学 管理学院
	XXX	教授	华中科技大学 管理学院
	X　X	教授	华中科技大学 管理学院

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Management

Stochastic Deep Learning Based Time Series Forecasting Techniques

Ph.D. Candidate : Zhang Xinze

Major : Management Science and Engineering

Supervisor : Prof. Bao Yukun

Huazhong University of Science and Technology

Wuhan 430074, P. R. China

April, 2023

独创性声明

本人声明所呈交的学位论文是我个人在导师的指导下进行的研究工作及取得的研究成果。尽我所知，除文中已标明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保　密 □，在 ____ 年解密后适用本授权书。

　　不保密 □。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

摘　　要

时间序列预测建模技术是能源、金融和公共卫生等众多应用领域的重要技术。针对时间序列预测建模问题，深度学习预测建模技术因其优异的预测性能成为了近年的研究热点。然而，以卷积神经网络和循环神经网络为代表的深度神经网络预测模型受限于梯度下降权重训练方法，需要在反复训练权重参数的基础上才可具备优异的预测性能，因此带来昂贵的计算消耗；加之深度学习预测模型复杂的结构参数设置，使得深度学习预测建模技术面临建模效率低与模型选择难等问题。随机映射方法作为神经网络模型的一种非迭代训练方法，通过固定模型随机初始化的输入层与隐藏层权重，利用简单直接的闭式求解算法计算模型输出层权重，以此完成权重参数训练过程，因而可有效提升神经网络模型的构造效率。

本学位论文基于随机映射方法对时间序列深度学习预测建模技术展开研究，针对随机深度神经网络预测模型在卷积结构和循环结构上的模型选择特有个性问题，提出随机卷积隐藏结构的构造优化方法和随机循环输出结构的构造优化方法；在此基础上探索随机深度神经网络特定代表结构的共性优化问题，提出随机深度神经网络预测模型一般结构下的二重特征选择方法和混合结构下的生长优化方法，进一步提升模型预测性能；在技术方法研究的同时，结合流感阳率预测、原油价格预测、电力负荷预测与电力价格预测等重要应用场景进行应用研究。本文的主要研究工作和创新性成果如下：

首先，针对卷积结构的随机深度神经网络预测模型构造与优化问题，提出基于误差反馈随机建模和贪心优化的新颖构造与优化方法，解决了既有方法的参数选择不足及输出权重病态矩阵难题。通过向隐藏层中递归添加随机卷积核，利用最小二乘法局部更新输出权重，构造出随机卷积隐藏结构，并证明了该构造方法的收敛性；借助贪心搜索优化卷积核参数，使其在单卷积层内自适应选择不同宽度的卷积核，具备了不同尺度局部特征的学习能力。在合成数据和多项现实数据上的实验表明，该模型具有极高的模型构造与选择效率，表现出与梯度下降训练的深度学习预测模型媲美甚至更优的预测性能。

其次，针对循环结构的随机深度神经网络预测模型构造与优化问题，提出了基于状态遮掩和粒子群优化的新颖构造与优化方法，解决了既有方法忽视的输出结构优化问题。基于对循环神经网络输出结构的随机映射实现，分析得出不同输出结构的异同优劣；通过向循环隐藏特征表示施加状态掩码，提出状态遮掩的随机循环输出结构，借助粒子群优化自适应选择循环隐藏特征表示和目标的映射取舍，提升了输出结构的学习能力。在人工合成、电力负荷和室外温度数据集上的实验表明，与既有随机循环神经网络及其输出结构优化方法相比，基于所提方法构造的模型具备明显的多步预测优势。

再次，针对一般结构的随机深度神经网络预测模型特征选择问题，提出了基于二重特征结构选择和树状结构帕森斯估计的新颖特征选择方法，消除了既有选择方法的一维输入特征结构局限。结合深度神经网络的时步多维度读取能力，利用滑动窗口构造时步多维度的二维输入特征结构；通过向每一输入时步添加时步掩码，建立基于时步维度与时步掩码组合的二重特征结构表示，借助树状结构帕森斯估计自适应优化特征结构参数，降低了抽象特征的学习难度。在人工合成、电力负荷和电力价格数据集上的实验表明，与既有卷积与循环结构随机深度神经网络及其特征选择方法相比，基于所提方法优化的模型具有更好的预测准确性与稳定性。

最后，针对混合结构的随机深度神经网络预测模型构造与优化问题，提出了基于误差反馈生长和三阶段优化的新颖混合结构构造与优化方法，消除了既有方法的单一结构局限。通过递归添加不同结构的随机子网络，利用岭回归局部更新输出权重，构造出鲁棒的混合隐藏结构，并证明了该构造方法的收敛性；通过预优化、子训练和调正则的三阶段优化方法对迭代添加的子网络进行完整优化，使其自适应选择出多种结构的子网络及参数，融合了不同深度结构的学习能力。在人工合成、空气污染和电力负荷数据集上的实验表明，与多种既有随机深度神经网络模型相比，基于所提方法构造的模型具有更好的预测性能。

关键词：时间序列预测；深度学习；随机映射；卷积神经网络；回声状态网络

Abstract

Time series forecasting is of great importance for a learning system in dynamic environments, playing a vital role in many real-world applications, such as energy, traffic, finance, and industry. Recent studies have shown that deep learning technique has shown intriguing prediction performance, leading to extensive research on the applications of the DNN models for time series forecasting. However, represented by the convolutional neural network and recurrent neural network models, the deep neural network–based forecasting models have complex architecture-related parameters and rely on gradient-based algorithms to train the weight-related parameters, making it extremely time-consuming and challenging to well establish a forecasting model. As an alternative method of training the neural network model, the stochastic mechanism fixes the weights of the input layer and the hidden layer after random initialization, and uses a simple and direct closed-form solution algorithm to calculate the weight of the output layer of the model, which can effectively improve the construction efficiency of the neural network model.

Therefore, this study investigates time series deep learning prediction modeling technology based on the stochastic mechanism, pays attention to the structure selection problems under the specific structures, and proposes the hidden structure construction method of the stochastic convolutional neural network and the output structure selection method of the stochastic recurrent neural network. On this basis, the optimization problems under the universal structure are explored, and the feature selection method as well as the parameter optimization method of the stochastic deep neural network prediction model are proposed to further improve the prediction performance of the models. At the same time, the application research is carried out in combination with important application scenarios, such as influenza prediction, crude oil price prediction, electricity load prediction, electricity price prediction, and so on. The main contributions of this dissertation are summarized as follows:

Focusing on the model construction problem of stochastic convolutional neural network–

based forecasting model, a novel error-feedback stochastic modeling strategy and greedy-based selection algorithm are proposed to craft the random convolutional neural network for time series forecasting. The proposed method suggests that random filters and neurons of the error-feedback fully connected layer are incrementally added to steadily compensate for the prediction error during the construction process, and then a greedy-based filter selection is introduced to enable the model to extract the different sizes of temporal features. Comprehensive experiments on the simulated dataset and several real-world datasets show that the proposed method exhibits stronger predictive power and lower computing overhead compared to trained state-of-the-art deep neural network models.

Focusing on the model construction problem of stochastic recurrent neural network–based forecasting model, a novel state mask strategy with particle swarm optimization is proposed to construct the random recurrent output structure for time series forecasting. Based on the investigation of the stochastic implementation of different recurrent output structures of training-based recurrent neural networks, the proposed method adds a mask to each step of the recurrent hidden features, and then a particle swarm optimization based mask selection is introduced to evolve the mapping relationships from recurrent hidden features to their targets, which improves the learning ability of the stochastic recurrent output architecture. Compared with the stochastic recurrent neural networks with the existing output architecture selection method, comprehensive experiments on the simulated dataset, electricity load dataset, and outside temperature dataset demonstrate the superiority of the proposed method.

Focusing on the input feature selection problem of stochastic deep neural network–based forecasting model, a novel dual feature-structured selection method with tree-structured parzen estimator is proposed. Based on the ability of deep neural architecture that can model multiple dimensions in each input time step, a multiple-step-dimension two-dimensional feature structure is established with a moving window schema. The proposed method adds a mask to each input step to represent the two-dimensional feature structure with the combination of step dimension and step mask, and then tree-structured parzen estimator is introduced to evolve the feature structure, which improves the learning ability of the stochastic deep

neural networks. Compared with the stochastic deep neural networks with the existing feature selection method, comprehensive experiments on the simulated dataset, electricity load dataset, and electricity price dataset demonstrate the superiority of the proposed method.

Focusing on the model construction problem of stochastic deep neural network–based forecasting model, a novel error-feedback triple-phase optimization strategy is proposed to grow stochastic deep neural network–based predictor with mixed deep neural architectures. The proposed method incrementally adds diverse deep subnetworks to the network, where the output weights of the subnetworks are calculated via ridge regression to improve the robustness of the constructed model, and the parameters of the subnetworks are evolved with pre-tuning, sub-tuning, and reg-tuning optimization, making the network take advantage of different deep neural architectures. Compared with the existing stochastic deep neural networks, comprehensive experiments on the simulated dataset, air pollution datasets and electricity load datasets demonstrate the superiority of the proposed method.

Keywords: Time series forecasting; deep learning; stochastic mechanism; convolutional neural network; echo state network

目　　录

1 时间序列预测建模框架

1.1 引言

时间序列预测建模技术是能源、交通、金融和工业生产等众多重要现实管理应用领域的重要支撑技术。在不同场景中，时间序列数据表现出动态机制不确定、模态表现不相同、数据规模不一致的高度复杂特征。因此，时间序列预测问题是一项颇具挑战和相当复杂的问题。如何针对不同场景构造准确、稳定和鲁棒的时间预测模型是众多研究者与工作者所关注的重要课题。

针对时间序列预测建模问题，各种预测建模技术推陈出新。其中，以深度神经网络为代表的深度学习预测建模技术因其优异的预测性能成为了近年的研究热点，涌现出大量基于深度神经网络模型的技术应用研究[1–9]。但在复杂多变的不同应用场景中，这些深度学习预测建模方法往往无法给予稳定的性能保证和一致的性能优势。原因在于，作为一种基于复杂神经网络结构构造的学习预测模型，深度学习预测模型往往具有远超传统机器学习方法或统计方法的参数设置，同时，传统深度学习预测模型的梯度下降迭代训练权重参数方法进一步降低了模型构造的效率，使得深度学习预测模型的构造过程面临建模效率低、模型选择难的问题。

作为深度学习预测建模技术应用与发展的关键，深度学习预测模型的高效构造与选择问题受到学术界与工业的广泛关注。为便捷构造深度神经网络模型，Google、Meta（Facebook）和 Microsoft 分别推出了 TensorFlow[10]、Pytorch[11] 和 CNTK[12] 等神经网络模型构造框架，通过封装常用深度神经网络基础结构（如卷积核结构、循环层结构和线性结构），集成多种先进梯度下降训练算法和矩阵运算与处理算法，为深度学习模型的深入研究与广泛应用提供了框架。为解决深度学习模型复杂的超参数搜索与优化问题，Bennet 等[13] 提出了一种开放优化框架 Nevergrad，通过内置贝叶斯优化、粒子群优化和差分进化等优化算法为深度学习模型参数搜索与优化提供了便利平台。Akiba 等[14] 基于贝叶斯类优化方法提供了一套运行定义（Define-by-run）模型的优化框架 OPTUNA，通过更个性化的搜索空间定义接口，支持搜索范围可根据采样结果自适应变动的复杂搜索设置。Liaw 等[15] 通过集成 Nevergrad 和 OPTUNA

等优化框架，提供了一套内置多种优化算法的分布式优化平台 Ray，进一步提高了模型参数优化的效率。

然而，这些建模与优化框架技术主要关注于广义建模技术中的特定环节，欠缺针对时间序列预测领域的建模考量。在面对复杂多变的时间序列管理应用预测需求中，仅依托于建模技术中的某一环节构造预测建模对于提升模型准确性、提高模型构造与选择效率、适应运营管理水平是不够的。因此，本章节基于前述各章在随机映射深度学习预测建模技术模型构造、隐藏结构优化、多输出结构优化和多输入特征选择等研究的基础上，设计出一套时间序列预测建模框架。通过整合既有的优秀神经网络建模框架与参数优化框架，针对时间序列预测问题，设计符合时间序列数据的数据定义与处理机制，完备包含时间序列预测建模的数据初始化、数据预处理、模型构造、模型优化和模型评价等构造与评价流程，集成多类别、多结构的现有对比预测方法，基于高度模块化和接口标准化的规范，使得用户能够较好地节约开发精力、复现已有工作、探索全新思路和解决预测问题。

1.2 框架需求分析

为应对不同场景下动态机制不确定、模态特征不相同、数据规模不一致的时间序列数据预测问题，基于数据驱动，构造自适应优化输入特征结构、模型隐藏结构与输出结构的时间序列预测模型，并呈现模型训练与预测效果的强弱程度，时间序列预测建模框架应当至少满足以下几点需求：

1.2.1 数据管理功能

深度学习预测建模技术的基础是大量的时间序列数据。时间序列数据高效与准确的处理以及统一的特征表示是构造普适结构深度学习预测模型的基础。然而，基于数据来源的场景与获取方式的不同，时间序列预测建模框架所面临的时间序列数据往往表现出不同的数据存储格式（如 json、txt、csv 和 xlxs 等文本格式）、时间粒度（如分钟、小时、日、周和月等）、数据缺失与异常情况（如 WTI 原油价格在 2020 年 4 月 20 日收盘时价格报-37.63）等数据表示。针对不同的时间序列预测建模技术，模型受入的时间序列数据结构也并不尽同，如本文第 5 章节所述，支持向量

机（SVM）与多层感知机（MLP）的多输入时间序列数据结构为包含样本数与特征维度的二维结构，而卷积神经网络（CNN）和循环神经网络（RNN）的多输入结构为包含样本数、时步维度和输入步长的三维结构。

因此，时间序列预测建模框架需要基于异构时间序列数据，提供差异化的数据读取功能，并针对不同类别的时间序列预测建模方法，提供差异的数据特征表示，而针对相同类别的预测建模方法，则需要提供统一的数据表示。同时，在此基础上需提供一致的数据预处理和后处理操作，以方便不同方法间的比较、展示与分析。

1.2.2 模型构造功能

构造时间序列预测模型是预测建模框架的核心功能。然而，受时间序列输入数据特征、模型自身参数和预测时长等因素的影响，预测模型在不同的时间序列预测任务上的表现可能有所差别。如本文第 4 章所述，节 4.3 中的实验结果表明，即使是同一预测任务上相同编码结构的回声状态网络（ESN）预测模型，受不同多输出结构的影响，多步预测性能亦表现出了明显差异。再如本文第 3 章所述，节 3.3 中的消融实验结果表明，通过对随机映射子网络的权重参数施加训练，可进一步提升模型预测性能。此外，为分析所构造预测模型的预测性能，丰富多样的对比预测模型是必要的，从而客观评价所构造模型的优劣。

因此，时间序列预测建模框架需要单元化、模块化和标准化的模型结构设计，以满足模型高度自定义的构造需求。同时，框架应包含多类别、多结构的优秀对比方法，为判断预测模型性能提供依据，从而适应不同预测任务的需要。

1.2.3 模型优化功能

解决时间序列预测模型的模型选择问题，优化所构造的预测模型是提升模型预测准确度从而增强模型应用性的关键。然而，不同预测模型建模方法的参数搜索优化空间各不相同，不同参数优化方法适应解决的优化问题也不相同。例如，作为群体智能优化算法的代表之一，粒子群优化（PSO）算法被应用于解决本文第 4 章节 ESN 预测模型的状态掩码优化问题，但不适用于解决第 5 章节所述的多输入特征结构优化问题。甚至，同一预测建模方法在不同预测任务上的参数搜索空间与优化算

法设置亦不相同。

因此，时间序列预测建模框架需要在标准化设计的模型接口基础上，提供多种优秀的优化算法，根据不同的时间序列预测任务情形，智能、高效和便捷地解决模型结构参数优化、多输出结构优化和多输入特征选择等模型选择问题，从而提高模型预测精度，进一步建立与任务情形相匹配的预测模型。

1.3 框架结构设计

根据时间序列预测任务的需要，融合既有模型构造与选择框架的优势[11,15]，本章节设计并开源提供了一套名为“OpenForecasting”的预测建模框架。该框架基于Python 这一动态语言，充分利用 Python 类的继承与多态特征，通过对 Pytorch 与 Ray 的封装，提供数据处理高度个性化、模型高度模组化、接口高度标准化的交互功能，基于完备的预测建模流程逻辑，有效地实现复杂时间序列数据的预测建模任务。这种框架结构有利于用户通过定义符合接口标准的模组，扩展现有方法或功能，满足更复杂的预测建模需求。

具体地，“OpenForecasting”预测框架主要通过 TaskWrapper 实现预测建模流程控制，利用 TaskLoader 实现数据载入与预处理，基于 TaskTuner 实现预测模型优化，辅以统一接口的模型库和优化方法库，完成预测建模。

1.3.1 预测流程控制

在本框架设计的 TaskWrapper 类中，基于接收用户指定的预测任务数据名称、预测时长、预测模型等预测任务信息，通过嵌套与调用数据加载、模型选择、模型预测和模型评价功能，自适应地完成时间序列预测模型的构造、优化与评价。

图 1.1 展示了“OpenForecasting”预测建模框架的流程控制主文件：task.py。通过接收用户传入的数据集名称、模型名称、预测时长、重复执行次数和预测准确度指标等任务信息，Task 类在初始化时完成如图 1.2 所示的数据预处理与加载、模型参数初始化和实验日志初始化过程。以本文第 4 章节中 ESN 模型在 MG 数据集上的预测建模实验为例，用户可采用如图 1.4 所示方式，通过自定义 task 主文件显式指定任务参数，构造基于 MG 数据集、预测时长为 17、重复实验次数为 20、预测评价

```python
from task.parser import
    get_parser
from task.TaskWrapper import Task

if __name__ == "__main__":

    args = get_parser()

    task = Task(args)
    task.tuning()
    task.conduct()
    task.evaluation(args.metrics)
```

图 1.1 预测建模框架的主文件：task.py

```python
from task.TaskLoader import Opt
import importlib
import ...

class Task(Opt):
    def __init__(self, args):

        self.data_config(args)
        self.model_config(args)
        self.exp_config(args)

    ...
```

图 1.2 TaskWrapper 中的 Task 类

```python
from task.parser import get_parser
from task.TaskWrapper import Task
if __name__ == "__main__":

    args = get_parser()

    args.cuda = True
    args.datafolder = 'paper.sm'
    args.dataset = 'mg'
    args.H = 17
    args.model = 'esn'
    args.rep_times = 20
    args.metrics = ['rmse','nrmse', 'mape']

    task = Task(args)
    task.tuning()
    task.conduct()
    task.evaluation(args.metrics)
```

图 1.3 自定义主文件 task.py 传入任务参数的示例

```
python task.py -datafolder paper.sm -dataset mg -H 17 -model esn -rep_times
    20 -metrics rmse nrmse mape
```

图 1.4 由命令行传入任务参数的示例

指标为 RMSE、NRMSE 和 MAPE 的 ESN 预测模型，并利用“task.tuning”函数完成 ESN 模型的参数优化，在优化结果的基础上更新模型预设参数，通过“task.conduct”函数进行预测建模，最终基于“task.evaluation”函数完成模型在各评价指标上的结果统计，获取对应指标的平均值和标准差等统计信息。用户亦可采用如图 1.4 所示的命令行传入参数方式，完成相同预测建模任务。此外，通过执行多个传入不同参数的 task.py 程序，用户可通过并行的方式针对同一预测建模任务构造不同的预测模型，

```python
class TaskDataset(Opt):
    def __init__(self, args):
        super().__init__()
        self.args = args
        self.info = Opt()
        self.info.normal = args.normal
        self.info.H = args.H
        self.info_config()
        self.sub_config()
        self.info.num_series = len(self.seriesPack)

    def info_config(self, ):
        pass

    def sub_config(self,):
        pass

    ...
```

图 1.5 TaskLoader 中的 TaskDataset 类

亦可基于相同的预测模型完成不同的预测建模任务，显著提高了用户使用的便利性。

1.3.2 数据预处理与加载

为应对不同数据存储结构的时间序列数据，适应不同输入步长设置、不同预测时长设置的时间序列预测任务需要，本框架通过设计 TaskLoader 中的 TaskDataset 父类，为不同时间序列数据提供加载接口。在转换得到统一数据格式的时间序列数据后，对数据进行归一化和数据集切分处理，完成预测建模任务的数据预处理与加载过程。

图 1.5 展示了“OpenForecasting”预测建模框架的 TaskDataset 父类，通过提供“info_config”与“sub_config”接口，使得用户可针对不同数据与任务情形，创建继承 TaskDataset 的 Data 子类，通过重构“info_config”与“sub_config”函数，灵活处理源数据以返回符合框架标准的加工数据。以本文第 3 章节中在 GEF 数据集上的预测建模实验为例，用户可采用如图 1.6 所示方式，创建继承 TaskDataset 的 gef_data 子类，在重构的“info_config”函数中指定了预测任务的归一化方法、输入步长、子时间序列数据名称等基础数据信息。继而，在重构的“sub_config”函数中，针对以“xlsx”文件格式存储的电力负荷数据，利用 pandas 数据分析工具[①]，构造出输入时步

[①]https://pandas.pydata.org.

```python
from task.TaskLoader import TaskDataset
import pandas as pd

class gef_data(TaskDataset):
    def __init__(self, args):
        super().__init__(args)


    def info_config(self):
        self.info.normal = False
        self.info.data_path = 'data/src/gef/2017_smd_hourly.xlsx'
        self.info.series_name = ['ME','NH','VT','CT','RI','SEMA','WCMA','NEMA']
        self.info.steps = 24*7

    def sub_config(self,):
        self.seriesPack = []

        for i, name in enumerate(self.info.series_name):
            df = pd.read_excel(self.info.data_path,sheet_name=name, index_col=None,header=0)
            raw_ts = df['RT_Demand'].values
            sub = self.pack_dataset(raw_ts)
            sub.index = i
            sub.name = name
            sub.merge(self.info)

            self.seriesPack.append(sub)
```

图 1.6 GEF 数据集数据预处理示例

为 168、提取出“ME”、“NH”、“VT”、“CT”、“RI”、“SEMA”、“WCMA”和“NEMA”共 8 个不同地区的电力负荷数据，基于 TaskDataset 父类提供的“pack_dataset”函数完成对子时间序列数据的预处理。

1.3.3 模型初始化与构造

预测模型的构造是预测框架中最为核心也是最为复杂的部分。为应对不同模型的不同参数设置与结构设置需求，高效兼容各类模型以完成参数初始化、模型构造与模型优化等流程，本框架通过基于结构模组化、输出标准化和参数接口化的设计思想，耦合模型参数接口和结构构造接口，统一预测模型输入输出标准，使得用户可针对不同需求灵活定义模型的相关参数或功能，实现模型的高效扩展，从而满足预测建模任务的需要。

```
import torch
import torch.nn as nn

class ESNCell(nn.Module):
    def __init__(self, init = 'vanilla', hidden_size = 500, input_dim = 1,
        nonlinearity = 'tanh', leaky_r = 1, weight_scale = 0.9, iw_bound = (-
        0.1, 0.1), hw_bound=(-1,1),  device='cpu'):
        super(ESNCell, self).__init__()
        self.input_dim = input_dim
        self.Hidden_Size = hidden_size
        ...

        self.ih = nn.Linear(self.input_dim, self.Hidden_Size, bias=False).to(
            self.device)
        self.hh = nn.Linear(self.Hidden_Size, self.Hidden_Size, bias=False).
            to(self.device)

        self.weight_init(self,)
        ...

    def forward(self, current_input, last_state):
        current_state = self.ih(current_input) + self.hh(last_state)
        current_state = self.act_f(current_state)
        _last_state = (1- self.leaky_r) * last_state + self.leaky_r *
            current_state

        return _last_state

    ...
```

图 1.7 回声状态网络单元模组示例

1.3.3.1 结构模组化

基于 Pytorch 框架自动差分、硬件加速和扩展性强等优势[11]，本框架对 Pytorch 框架提供的神经网络单元结构进行了封装，建立出常用的深度神经网络结构模组，进一步增强了用户构造深度神经网络预测模型的可读性、便利性与扩展性。

例如，图 1.7 展示了基于 Pytorch 框架实现的 ESN 单元模组。而后，调用 ESN 单元模组循环读取时间序列输入特征，实现出如图 1.8 所示的 ESN 编码过程模组。类似的，以本文第 5 章中所构造的回声状态卷积结构和卷积回声状态结构为例，通过如图 1.9 所示的简洁模组调用方式，即可建立出耦合卷积结构和回声状态结构的神经网络混合结构。通过本框架对深度神经网络结构的模组化，用户得以快捷构造出所需模型。

```
1  import torch
2  import torch.nn as nn
3
4  class esnLayer(nn.Module):
5      def __init__(self, init = 'vanilla', hidden_size = 500, input_dim = 1,
           nonlinearity = 'tanh', leaky_r = 1, weight_scale = 0.9, iw_bound = (-
           0.1, 0.1), hw_bound=(-1,1),  device='cpu'):
6          super(esnLayer, self).__init__()
7          self.esnCell = ESNCell(init=init,hidden_size=hidden_size,input_dim=
               input_dim,nonlinearity=nonlinearity,leaky_r=leaky_r,weight_scale=
               weight_scale,iw_bound=iw_bound,hw_bound=hw_bound,device=device)
8
9
10     def forward(self, x, _last_state = None):
11         with torch.no_grad():
12             samples, time_steps = x.shape[0], x.shape[2]
13             layer_hidden_state = torch.empty(samples,self.esnCell.Hidden_Size
                   , time_steps).to(self.esnCell.device)
14
15             last_state = torch.zeros(samples, self.esnCell.Hidden_Size).to(
                   self.esnCell.device) if _last_state is None else _last_state.
                   detach().clone().to(self.esnCell.device)
16
17             for t in range(time_steps):
18                 last_state = self.esnCell(x[:,:,t],  last_state)
19                 layer_hidden_state[:,:,t] = last_state
20
21         return layer_hidden_state, last_state
22
23     ...
```

图 1.8 回声状态网络编码过程模组示例

1.3.3.2 输出标准化

尽管不同预测建模技术所构造出的模型结构可能有较大差别，但在相同的预测任务下，这些模型的预测输出结果应当具有相同的数据结构形式。基于此，本框架通过统一预测模型的拟合过程接口，在不同预测模型方法间建立出一种标准化的输入输出规范，从而为框架的模型优化与评价功能奠定基础。

具体地，本框架通过 Python 类的形式构造具有标准输出形式的不同预测模型。以图 1.10 所展示的 ESN 模型为例，具体说明本框架的模型输出标准化思想。对于本框架所实现的预测模型，在类初始化“__init__”阶段，均会被赋予一个记录模型拟合信息的属性“fit_info”。而后，通过“xfit”函数，模型将读取数据预处理与加载环节切分出的训练集与验证集数据（对应图 1.10 中第 15 与 16 行），用于训练模型相关参数和验证模型效果（对应图 1.10 中第 17 至 24 行），并将对应的训练集误差与验证集误差保存在“fit_info”中（对应图 1.10 中第 26 与 17 行），供流程控制程序进行下

```
class cesLayer(nn.Module):
    def __init__(self, cnn_hyper, esn_hyper):
        super(cesLayer, self).__init__()
        self.cnnLayer = cnnLayer(**cnn_hyper.dict)
        self.esnLayer = esnLayer(**esn_hyper.dict)

    def forward(self, x)
        fm, _ = self.cnnLayer(x)
        layer_hidden_state, last_state = self.esnLayer(fm)
        return layer_hidden_state, last_state

    ...
```

(a) 卷积回声状态结构代码示例

```
class escLayer(nn.Module):
    def __init__(self, cnn_hyper, esn_hyper):
        super(cesLayer, self).__init__()
        self.cnnLayer = cnnLayer(**cnn_hyper.dict)
        self.esnLayer = esnLayer(**esn_hyper.dict)

    def forward(self, x)
        layer_hidden_state, _ = self.esnLayer(x)
        fm, fm_flatten = self.cnnLayer(layer_hidden_state)
        return fm, fm_flatten

    ...
```

(b) 回声状态卷积结构代码示例

图 1.9 基于结构模组化设计的深度神经网络混合结构代码示例

一步的优化或评价工作。

1.3.3.3 参数接口化

如图 1.11 所示，本框架通过设计模型参数设置的 Model_base 父类，为框架内的每个模型提供 hyper、tuner 和 tuning 三种属性以及“base_init”、“setting_init”和“setting_modify”三种函数接口，将模型的结构超参数设置、优化器参数设置与参数优化结果进行单独管理。

图 1.12 展示了基于 Model_base 父类的 esn_base 子类，通过重构 Model_base 中的“base_init”函数，将该子类与对应路径下存储的 ESN 模型结构类进行匹配；通过调用与重构“setting_init”函数，为 ESN 模型提供了隐藏层数量为 400、激活函数类别为“sigmoid”、输入与隐藏层权重随机初始化分布为 $[-0.1, 0.1]$ 和 $[-1, 1]$、隐藏状

```python
import torch.nn as nn
from task.metric import rmse
import ...

class EchoStateNetwork(nn.Module):
    def __init__(self, opts=None, logger=None):
        super(EchoStateNetwork, self).__init__()
        self.opts = opts
        self.logger = logger

        self.fit_info = Opt()
        ...

    def xfit(self, train_data, val_data):
        train_loader = self.data_loader(train_data)
        val_loader = self.data_loader(val_data)
        h_states, x, y = self.batch_transform(train_loader)

        self.update_readout(h_states, x, y)

        pred = self.readout(self.io_check(h_states, x)[:,:,-1])

        _, y, pred = self.loader_pred(train_loader)
        _, val_y, vpred = self.loader_pred(val_loader)

        self.fit_info.trmse = rmse(y,pred)
        self.fit_info.vrmse = rmse(val_y, vpred)

        return self.fit_info

    ...
```

图 1.10 ESN.py 文件中的 EchoStateNetwork 类示例

```python
class Model_base(Opt):
    def __init__(self):
        super().__init__()

        self.hyper = Opt()
        self.tuner = Opt()
        self.tuning = Opt()

        self.base_init()
        self.setting_init()
        self.setting_modify()

    ...
```

图 1.11 Model_base 类示例

态读取步长为 2 的基本参数。通过这种方法，用户可以通过在子类中修改“base_init”函数，使得多个不同模型共享一套参数设置，从而更好观测相同参数设置不同模型结构的预测模型性能差异。用户亦可通过共享父类中的“base_init”和“setting_init”

```
class esn_base(Model_base):
    def base_init(self,):
        self.import_path='models/stochastic/esn/ESN.py'
        self.class_name = 'EchoStateNetwork'

    def setting_init(self,):
        self.hyper.input_dim = 1
        self.hyper.hidden_size = 400
        self.hyper.nonlinearity = 'sigmoid'
        self.hyper.iw_bound = (-0.1, 0.1)
        self.hyper.hw_bound = (-1, 1)
        self.hyper.readout_steps = 2
        ...

    ...
```

图 1.12 继承 Model_base 类的 esn_base 子类

```
class fcd_esn(esn_base):

    def setting_modify(self,):
        self.hyper.readout_steps
            = 1

    ...
```

(a) 继承 esn_base 类的 fcd_esn 子类示例

```
class stk_esn(esn_base):

    def setting_modify(self,):
        self.hyper.readout_steps
            = 168 / 2

    ...
```

(b) 继承 esn_base 类的 stk_esn 子类示例

图 1.13 基于 esn_base 类构造的不同多输出结构模型参数示例

```
from ray import tune
from task.ModelSetting import esn_base

class fcd_esn(esn_base):
    def __init__(self):
        super().__init__()
        self.tuner.algo = 'tpe'
        self.tuner.iters = 200
        self.tuner.resource = {
            'cpu': 5, 'gpu': 0.5
        }
        self.tuning.iw_bound = tune.loguniform(1e-5, 1e-1)
        self.tuning.weight_scaling = tune.uniform(0.2, 0.99)
        self.tuning.nonlinearity = tune.choice(['sigmoid', 'tanh'])
        self.tuning.hidden_size = tune.qrandint(500, 1500, 50)

    ...
```

图 1.14 优化器资源与搜索参数设置示例

```python
import ray,importlib
from ray import tune
from ray.air import session
from ray.tune.search.optuna import OptunaSearch
import ...

class TaskTuner(Opt):
    def __init__(self, opts, subPack, logger):
        super().__init__()
        self.merge(opts)
        self.points_to_evaluate = []
        if 'points_to_evaluate' in self.tuner.dict:
            self.points_to_evaluate = self.tuner.points_to_evaluate

        self.train_data = subPack.train_data
        self.valid_data = subPack.valid_data
        self.batch_size = subPack.batch_size
        self.metric = 'vrmse'
        self.logger = logger
        self.algo_init()

    def algo_init(self,):
        tpe_search = OptunaSearch(metric=self.metric,mode='min',
            points_to_evaluate=self.points_to_evaluat)
        self.algo_func = tpe_search()

    def fitness(self, config):
        _hyper = Opt(init=self.hyper)
        _hyper.update(config)

        model = importlib.import_module(self.import_path)
        model = getattr(model, self.class_name)
        model = model(_hyper, self.logger)
        fit_info = model.xfit(self.train_data, self.valid_data)
        trmse, vrmse = fit_info.trmse, fit_info.vrmse

        session.report({'trmse': trmse,'vrmse': vrmse,})

    def conduct(self,):
        ray.init()
        ray_tuner = tune.Tuner(
            tune.with_resources(tune.with_parameters(self.fitness),resources=
                self.tuner.resource),
            param_space=self.tuning.dict,
            tune_config=tune.TuneConfig(search_alg=self.algo_func,num_samples
                =self.tuner.iters),
            )

        results = ray_tuner.fit()
        best_result = results.get_best_result(self.metric, 'min')
        return best_result.config
```

图 1.15 TaskTuner 类示例

函数，自定义“setting_modify”函数，实现相同模型结构不同模型参数的预测模型，从而满足预测建模实验分析的需要。以本文第 4 章节中构造基于 GEF 数据集的全连接解码器（FCD）和堆栈结构（STK）ESN 预测模型的实验为例，如图 1.13 所示，通

过重构“setting_modify”函数调整隐藏状态读取步长，利用 Python 语言中的多态形式，便捷构造出了 FCD-ESN 和 STK-ESN 预测模型。

此外，用户通过设置或修改 tuner 和 tuning 相关参数，可灵活调整优化器的计算资源分配与搜索空间，从而优化利用当前计算资源条件，满足差异化的建模任务要求。具体地，图 1.14 展示了一种优化器资源与搜索空间参数的设置示例。结合本文第 4 章中基于 AMD 5950X CPU（16 核 32 线程）与三卡 Nvidia RTX A4000 GPU 的计算条件，tuner 的参数表明：优化算法为 TPE 算法，TPE 算法最多采样次数为 200，优化过程以并行计算的方式加速进行，单个模型程序占用 5 个 CPU 线程与半张 GPU 计算卡资源，模型并发数为 6；tuning 的参数指代：ESN 模型输入权重分布参数的搜索边界为 $[0.00001, 0.1]$，隐藏层权重缩放参数搜索边界为 $[0.2, 0.99]$，激活函数的选择范围为 $\{sigmoid, tanh\}$，隐藏层神经元数量的选择范围为步进 50，边界为 500 和 1500 的数列 $\{500, 550, 600, \ldots, 1500\}$。

1.3.4 模型选择与优化

利用 Ray 框架所提供的丰富优化算法与并行计算机制，在前述设计思想的基础上，本框架对 Ray 框架提供的优化接口结合预测建模任务背景进行了封装，建立出适用于解决时间序列预测建模技术模型选择问题的优化平台，从而为不同预测模型提供模型选择与优化功能，进一步满足用户对于模型预测性能的要求。具体地，本章节通过如图 1.15 所示的优化器 TaskTuner 示例，详细说明本框架的模型选择与优化过程。

在 TaskTuner 类的“__init__”阶段，通过接收参数“opts”，数据集包“subPack”和实验日志“logger”完成优化器的初始化。其中，“opts”即为如图 1.14 所示构造的模型参数类，通过图 1.15 中第 12 行所示的”merge” 操作，将模型的 hyper、tuner 与 tuning 等参数信息传递给 TaskTuner 类，从而确定模型初始参数、优化器参数和参数搜索空间等设置。“subPack”为如图 1.5 所示经预处理的数据文件，从而传入模型训练与验证所需的训练集与验证集数据。此外，如图 1.5 中第 22 行至 26 行，TaskTuner 类通过调用“algo_init”函数，完成以优化算法的初始化。显然，本框架用户可以通过重构“algo_init”函数，灵活修改或扩展优化算法种类，从而满足不同优化场景的

表 1.1 “OpenForecasting”框架提供的预测模型

模型名称	参考文献
Naive	Box et al. Time Series Analysis: Forecasting and Control[16].
ARIMA	Hyndman et al. Automatic Time Series Forecasting: The Forecast Package for R[17].
HW	Chatfield, Chris. The Holt-Winters Forecasting Procedure[18].
MSVR	Bao et al. Multi-Step-Ahead Time Series Prediction Using Multiple-Output Support Vector Regression[19].
MLP	Bao et al. PSO-MISMO Modeling Strategy for MultiStep-Ahead Time Series Prediction[20].
CLPSO-ML	Yang et al. Comprehensive Learning Particle Swarm Optimization Enabled Modeling Framework for Multi-Step-Ahead Influenza Prediction[21].
Elman RNN	Hewamalage et al. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions[22].
GRU	Cho et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches[23].
LSTM	Hochreiter et al. Long Short-Term Memory[24].
DeepAR	Salinas et al. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks[25].
CNN	Koprinska et al. Convolutional Neural Networks for Energy Time Series Forecasting[26].
ConvRNN	Livieris et al. A CNN–LSTM Model for Gold Price Time Series Forecasting[27].
RVFL	Igelnik et al. Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link net[28].
IELM	Huang et al. Extreme Learning Machine: Theory and Applications[29].
SCN	Wang et al. Stochastic Configuration Networks: Fundamentals and Algorithms[30].
ESN	Jaeger, Herbert. The “Echo State” Approach to Analysing and Training Recurrent Neural Networks with an Erratum Note[31].
DESN	Gallicchio et al. Echo State Property of Deep Reservoir Computing Networks[32].
GESN	Qiao et al. Growing Echo-State Network with Multiple Subreservoirs[33].
PSO-GESN	Li et al. PSO-Based Growing Echo State Network[34].
Stoc-CNN	Yu et al. Impact of Random Weights on Nonlinear System Identification Using Convolutional Neural Networks[35].
ESM-CNN	Zhang et al. Error-Feedback Stochastic Modeling Strategy for Time Series Forecasting with Convolutional Neural Networks[36].

表 1.2 “UniV-Forecasting”框架支持的优化算法

算法名称	参考文献
RS	Bergstra et al. Random Search for Hyper-Parameter Optimization[37].
BO	Snoek et al. Practical Bayesian Optimization of Machine Learning Algorithms[38].
BOHB	Falkner et al. BOHB: Robust and Efficient Hyperparameter Optimization at Scale[39].
CFO	Wu et al. Frugal Optimization for Cost-Related Hyperparameters[40].
BS	Wang et al. Economic Hyperparameter Optimization with Blended Search Strategy[41].
PSO	Kennedy et al. Particle Swarm Optimization[42].
TPE	Bergstra et al. Algorithms for Hyper-Parameter Optimization[43].

需要。

TaskTuner 类的优化过程执行阶段，如图 1.5 中第 41 行至 50 行，利用前述输出标准化与参数接口化的设计思想与规范，通过对模型初始化与构造的封装，确定优化控制器 ray_tuner 的适应度“fitness”函数，通过模型 tuner 参数中的相关设置，完成 ray_tuner 的初始化。在如图 1.5 中第 48 行所示的优化过程中，基于模型 tuning 属性所定义的参数空间“param_space”（对应图 1.5 中第 44 行），ray_tuner 会对被优化的参数进行采样或更新得到新的参数解，这些解以“fitness”函数的“config”形式传递给模型的既有参数集合“_hyper”，并对“_hyper”进行更新，而后基于更新后的参数集合构造预测模型，通过模型“xfit”函数所界定的标准化输入输出形式，得到模型在当前参数下的训练集误差与验证集误差。这些误差被汇报给 ray_tuner 以进行下一轮参数采样或更新过程，在完成最大迭代次数后，返回模型的最优参数。

基于本节所述的框架结构设计方式，如表 1.1 和表 1.2 所示，本框架提供了多种预测建模技术与方法，使得用户可在预测建模实验中对比分析所提模型与既有模型间的性能差异，找寻不同预测模型结构或智能优化方法在所选预测任务上的优劣，从而启发出更多预测建模方法，进一步提高解决时间序列预测建模问题的能力与水平。

1.4 框架逻辑流程

图 1.16 展示了本框架的逻辑流程图。该流程图中的步骤说明如下：

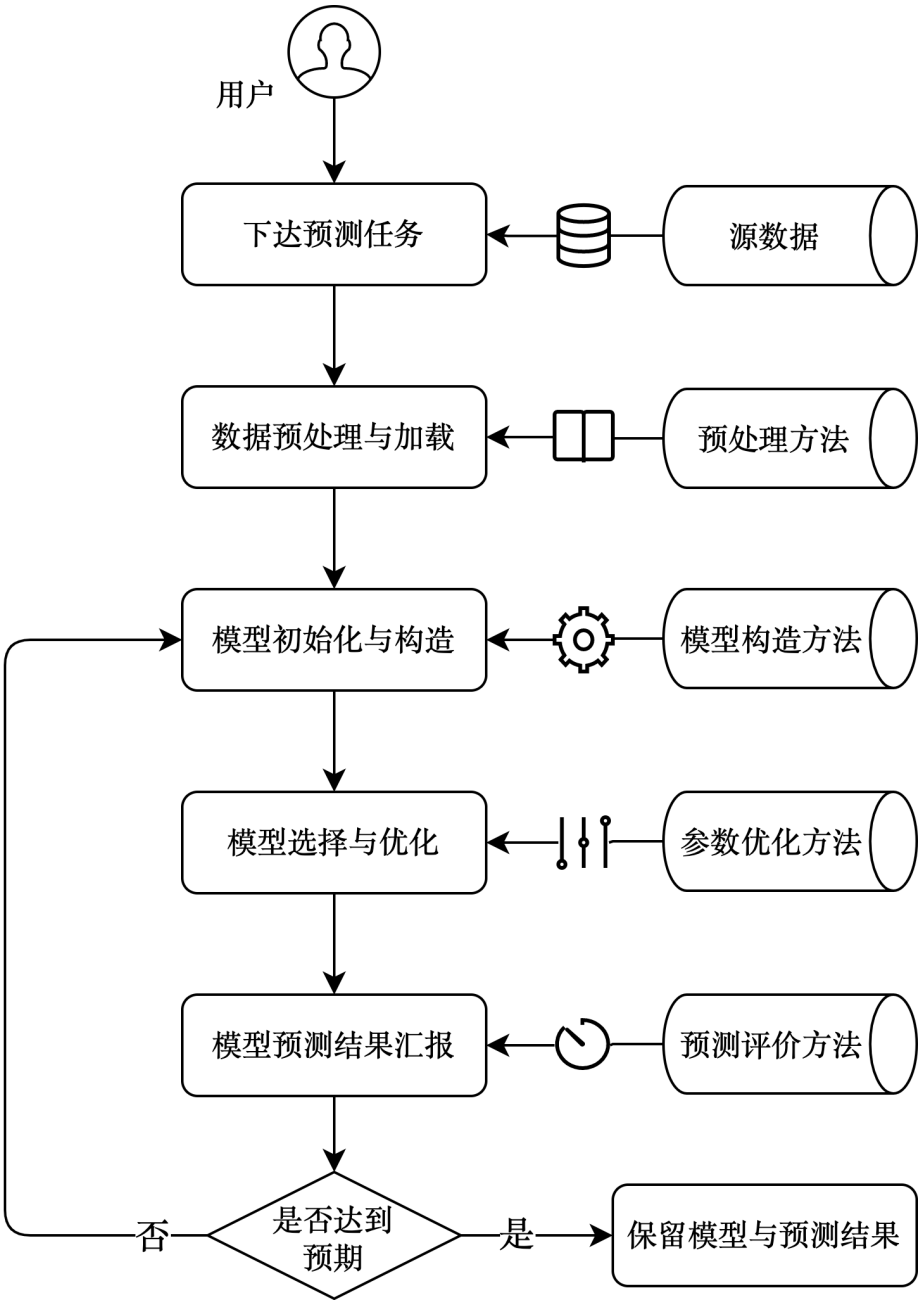

图 1.16 框架逻辑流程图

（1）用户下达预测任务。基于不同的时间序列预测任务情形，用户收集到所需的时间序列源数据。这些源数据可以是以“txt”、“xlsx”、“csv”和“json”等格式保存的本地文件，也可以是存储在本地或云端数据库系统中的数据库文件。

（2）数据预处理与加载。基于所收集到的源数据，定义继承 TaskDataset 父类的数据文件 Data 子类，通过重构“info_config”函数完成时间序列预测任务的基础数

据信息设定，利用“sub_config”函数抽取出预测任务所需的时间序列数据，并结合与“info_config”中的信息设定相匹配的预处理方法，完成时间序列数据的异常值处理、归一化、数据集切分等预处理工作，打包预处理数据与对应信息设定并予以加载。

（3）模型初始化与构造。基于使用本框架目标的不同，用户可从内置预测模型方法中选择出合适的模型或自行构造新的模型。针对所选择或设计的模型构造方法，定义继承 Model_base 父类的模型参数 Model 子类，通过修改“base_init”函数指定模型构造方法的保存路径与模型类名，通过修改“setting_init”或“setting_modify”函数指定模型参数的初始化设置、模型搜索空间设置和优化器参数设置等参数信息。而后，框架将基于已加载的数据信息和已指定的模型参数信息，自动完成模型的初始化与构造工作。

（4）模型选择与优化。基于已加载的训练集与验证集数据以及已指定的模型参数搜索空间和优化器参数等设置，框架将自动分配计算资源，调用优化器高效进行模型选择过程，完成模型参数的优化。该优化过程不仅包含模型隐藏层神经元数量、激活函数种类和隐藏层层数等模型结构参数，通过本文第 5 章节所提出的二重特征结构及其编码表示方法，多输入特征选择与优化亦被囊括在内。同时，用户可重构 TaskTuner 类的“algo_init”函数，灵活选取不同的优化算法，进一步提高模型预测性能。

（5）模型预测结果汇报。基于已加载的测试集数据和已优化的预测模型，框架将基于所设置的重复试验次数与指定的预测准确度指标，自动完成模型测试，并整理模型在当前设置下的预测结果，详细汇报出各预测准确度指标的平均值、标准差、时步误差曲线等信息，同时提供 Naive 方法的测试结果以做基准参考。若未达预期，则可通过修改模型结构设计、参数搜索空间和优化算法选择等设置，继续调整模型直到符合预期，最终保留模型与预测结果。

1.5 小结

面对复杂多变的时间序列预测需求，为提升时间序列预测建模技术的集成水平与运行效率，本章节在已有优秀模型构造框架与参数优化框架的基础上，设计并开

源了一套兼具高集成度与强扩展性的时间序列预测建模框架“OpenForcasting”。通过对时间序列预测建模框架的用户需求分析，归纳出包含数据管理功能、模型构造功能与模型优化功能在内的核心需求功能，完成了时间序列预测建模框架的系统结构设计与逻辑流程设计。基于此，本框架内置了不同类别的多种优秀预测模型方法，通过良好的接口设计规范，使得用户能够灵活处理不同结构的时间序列源数据，设计模型结构，调整模型参数和选择优化算法，从而更好解决时间序列预测问题。

致 谢

在博士研究生经历中，有过许多挑战和成长。回想一路走来，有过研究时的迷惘，有过发表时的兴奋，也有过展望时的徘徊，感慨万千。在此，向所有的支持者表示感谢。

.....

参考文献

[1] Y.-X. Wu, Q.-B. Wu, J.-Q. Zhu. Improved EEMD-Based Crude Oil Price Forecasting Using LSTM Networks. Physica A: Statistical Mechanics and its Applications, 2019, 516: 114–124

[2] Y. Zhao, J. Li, L. Yu. A Deep Learning Ensemble Approach for Crude Oil Price Forecasting. Energy Economics, 2017, 66: 9–16

[3] O. B. Sezer, M. U. Gudelek, A. M. Ozbayoglu. Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005–2019. Applied Soft Computing, 2020, 90: 106181

[4] M. Cai, M. Pipattanasomporn, S. Rahman. Day-Ahead Building-Level Load Forecasts Using Deep Learning vs. Traditional Time-Series Techniques. Applied Energy, 2019, 236: 1078–1088

[5] K. B. Lindberg, P. Seljom, H. Madsen, et al. Long-Term Electricity Load Forecasting: Current and Future Trends. Utilities Policy, 2019, 58: 102–119

[6] H. Shi, M. Xu, R. Li. Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN. IEEE Transactions on Smart Grid, 2018, 9(5): 5271–5280

[7] N. Laptev, J. Yosinski, L. E. Li, et al. Time-Series Extreme Event Forecasting with Neural Networks at Uber. in: International Conference on Machine Learning. 2017: 1–5

[8] J. Wang, Q. Gu, J. Wu, et al. Traffic Speed Prediction and Congestion Source Exploration: A Deep Learning Method. in: IEEE 16th International Conference on Data Mining. 2016: 499–508

[9] J. Xiao, Z. Xiao, D. Wang, et al. Short-Term Traffic Volume Prediction by Ensemble Learning in Concept Drifting Environments. Knowledge-Based Systems, 2019, 164: 213–225

[10] M. Abadi, P. Barham, J. Chen, et al. Tensorflow: A System for Large-Scale Machine Learning. in: 12th USENIX Symposium on Operating Systems Design and Implementation. 2016: 265–283

[11] A. Paszke, S. Gross, F. Massa, et al. Pytorch: An Imperative Style, High-Performance Deep Learning Library. in: Advances in Neural Information Processing Systems. 2019: 8026–8037

[12] F. Seide, A. Agarwal. CNTK: Microsoft's Open-Source Deep-Learning Toolkit. in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016: 2135–2135

[13] P. Bennet, C. Doerr, A. Moreau, et al. Nevergrad: Black-Box Optimization Platform. ACM SIGEVOlution, 2021, 14(1): 8–15

[14] T. Akiba, S. Sano, T. Yanase, et al. Optuna: A next-Generation Hyperparameter Optimization Framework. in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 2623–2631

[15] R. Liaw, E. Liang, R. Nishihara, et al. Tune: A Research Platform for Distributed Model Selection and Training. in: International Conference on Machine Learning. 2018

[16] G. E. Box, G. M. Jenkins, G. C. Reinsel. Time Series Analysis: Forecasting and Control. 2015

[17] R. J. Hyndman, Y. Khandakar. Automatic Time Series Forecasting: The Forecast Package for R. Journal of Statistical Software, 2008, 27: 1–22

[18] C. Chatfield. The Holt-Winters Forecasting Procedure. Journal of the Royal Statistical Society: Series C (Applied Statistics), 1978, 27(3): 264–279

[19] Y. Bao, T. Xiong, Z. Hu. Multi-Step-Ahead Time Series Prediction Using Multiple-Output Support Vector Regression. Neurocomputing, 2014, 129: 482–493

[20] Y. Bao, T. Xiong, Z. Hu. PSO-MISMO Modeling Strategy for MultiStep-Ahead Time Series Prediction. IEEE Transactions on Cybernetics, 2014, 44(5): 655–668

[21] S. Yang, Y. Bao. Comprehensive Learning Particle Swarm Optimization Enabled Modeling Framework for Multi-Step-Ahead Influenza Prediction. Applied Soft Computing, 2021, 113: 107994

[22] H. Hewamalage, C. Bergmeir, K. Bandara. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions. International Journal of Forecasting, 2021, 37(1): 388–427

[23] K. Cho, B. van Merriënboer, D. Bahdanau, et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. Proceedings of SSST EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, 2014, : 103–111

[24] S. Hochreiter, J. Schmidhuber. Long Short-Term Memory. Neural Computation, 1997, 9(8): 1735–1780

[25] D. Salinas, V. Flunkert, J. Gasthaus, et al. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. International Journal of Forecasting, 2020, 36(3): 1181–1191

[26] I. Koprinska, D. Wu, Z. Wang. Convolutional Neural Networks for Energy Time Series Forecasting. in: International Joint Conference on Neural Networks. 2018: 1–8

[27] I. E. Livieris, E. Pintelas, P. Pintelas. A CNN–LSTM Model for Gold Price Time-Series Forecasting. Neural Computing and Applications, 2020, 32: 17351–17360

[28] B. Igelnik, Y.-H. Pao. Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link Net. IEEE Transactions on Neural Networks, 1995, 6(6): 1320–1329

[29] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew. Extreme Learning Machine: Theory and Applications. Neurocomputing, 2006, 70(1): 489–501

[30] D. Wang, M. Li. Stochastic Configuration Networks: Fundamentals and Algorithms. IEEE Transactions on Cybernetics, 2017, 47(10): 3466–3479

[31] H. Jaeger. The "Echo State" Approach to Analysing and Training Recurrent Neural Networks-with an Erratum Note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report, 2001, 148(34): 13

[32] C. Gallicchio, A. Micheli. Echo State Property of Deep Reservoir Computing Networks. Cognitive Computation, 2017, 9(3): 337–350

[33] J. Qiao, F. Li, H. Han, et al. Growing Echo-State Network with Multiple Subreservoirs. IEEE Transactions on Neural Networks and Learning Systems, 2017, 28(2): 391–404

[34] Y. Li, F. Li. PSO-Based Growing Echo State Network. Applied Soft Computing, 2019, 85: 105774

[35] W. Yu, M. Pacheco. Impact of Random Weights on Nonlinear System Identification Using Convolutional Neural Networks. Information Sciences, 2019, 477: 1–14

[36] X. Zhang, K. He, Y. Bao. Error-Feedback Stochastic Modeling Strategy for Time Series Forecasting with Convolutional Neural Networks. Neurocomputing, 2021, 459: 234–248

[37] J. Bergstra, Y. Bengio. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 2012, 13(2)

[38] J. Snoek, H. Larochelle, R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. in: Advances in Neural Information Processing Systems. 2012: 2960–2968

[39] S. Falkner, A. Klein, F. Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. in: International Conference on Machine Learning. 2018: 1437–1446

[40] Q. Wu, C. Wang, S. Huang. Frugal Optimization for Cost-Related Hyperparameters. in: Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence. 2021: 10347–10354

[41] C. Wang, Q. Wu, S. Huang, et al. Economic Hyperparameter Optimization with Blended Search Strategy. in: International Conference on Learning Representations. 2020

[42] J. Kennedy, R. Eberhart. Particle Swarm Optimization. in: International Conference on Neural Networks. 1995: 1942–1948

[43] J. Bergstra, R. Bardenet, Y. Bengio, et al. Algorithms for Hyper-Parameter Optimization. in: Advances in Neural Information Processing Systems. 2011: 2546–2554

附录 1　攻读博士学位期间取得的研究成果

已发表论文：

[1] **Xinze Zhang**, Kun He, and Yukun Bao. Error-feedback Stochastic Modeling Strategy for Time Series Forecasting with Convolutional Neural Networks. Neurocomputing, 2021, 459:234-248.(SCI 源刊，IF 5.719，署名单位: 华中科技大学)

[2] Jianhua Yang, **Xinze Zhang**, and Yukun Bao. Short-term Load Forecasting of Central China based on DPSO-LSTM. In Proceedings of IEEE 4th International Electrical and Energy Conference, IEEC 2021. (EI 会议，署名单位: 华中科技大学)

[3] **Xinze Zhang**, Junzhe Zhang, Zhenhua Chen, and Kun He. Crafting Adversarial Examples for Neural Machine Translation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, ACL 2021. (CCF A 类国际会议，署名单位: 华中科技大学)

工作论文：

[1] **Xinze Zhang**, Kun He, Yukun Bao, and Qi Sima. Error-feedback Triple-phase Optimization to Configurable Convolutional Echo State Network for Time Series Forecasting.

[2] **Xinze Zhang**, Qi Sima, Kun He, Yukun Bao, and Shuhan Chen. Enhancing Echo State Network with Particle Swarm Bayesian Optimization Enabled Echo State Selection for Time Series Forecasting.

[3] Qi Sima, **Xinze Zhang**, and Yukun Bao. Reinforced Decoder: Towards Training Sequence-to-Sequence Model for Time Series Forecasting.

附录 2　已发表论文与博士学位论文的关系

序号	成果名称	成果形式	成果主要内容	与学位论文 对应关系
1	Error-feedback Stochastic Modeling Strategy for Time Series Forecasting with Convolutional Neural Networks	SCI 期刊 （已发表）	提出了误差反馈随机建模的迭代构造策略，结合贪心搜索自适应确定卷积结构，构造出一种新颖、高效且准确的随机卷积神经网络预测模型	该成果为论文第 3 章主要内容
2	Short-term Load Forecasting of Central China based on DPSO-LSTM	EI 会议 （已发表）	提出了时步多维度的时序输入特征二维结构，结合粒子群优化算法选择时步维度与模型参数，提升了 LSTM 模型的短期电力负荷预测效果	基于对该成果的改进，构成论文第 5 章主要内容

附录 3　攻读博士学位期间参与的科研项目

1. 国家自然科学基金面上项目：大数据环境下基于计算智能的预测建模技术及其在电力负荷预测中的应用，批准年限：2019/01 - 2022/12。
2. 国家自然科学基金面上项目：自然语言处理深度模型的对抗攻防关键算法研究，批准年限：2021/01 - 2024/12。
3. 湖北省国际科技合作项目：深度学习模型对抗攻防基础理论与算法研究，批准年限：2021/09 - 2023/09。
4. 国家电网公司科技项目：华中区域共享型电力交易与服务平台关键技术研究，批准年限：2020/06 - 2021/11。
5. 国家电网公司科技项目：渝鄂直流运行条件下华中消纳西南水电交易电量库模型建立与效益分析，批准年限：2019/05 - 2020/11。