基于高性能海量数据处理算法的短期负荷预测研究

专业：电气工程及其自动化

学生：尹思凯 指导老师：刘洋

摘 要

当下智能电网建设与电力市场改革对电力系统的运行经济性和调度精细性提出了更高的要求，同时随着现代用电信息采集系统和智能电表的普及，电力系统所产生的数据规模愈发增加，电力大数据背景愈发凸显，对短期负荷预测等调度关键环节提出了海量数据处理的高性能要求。本文基于电力系统数据科学领域的前沿研究成果，提出基于高性能海量数据处理算法的电力系统短期负荷预测方案，通过引入深度学习模型和分布式并行计算框架提升预测的精度和用时性能，并以不同规模尺度的实际历史负荷数据集进行实验验证。本毕业论文主要内容有：

首先，为了提升模型预测精度性能，本文构建了深度学习中的长短期记忆循环神经网络模型（LSTM）用于海量数据背景下的短期电力负荷预测，并在主流的 Google TensorFlow 机器学习框架以 Python 编程实现该 LSTM 神经网络负荷预测模型，通过超参数调整优化整定模型网络结构，并分别在区域聚合负荷和单表居民用户负荷两种不同尺度的数据集进行上进行预测，将模型预测精度与传统日前负荷预测算法 BP 神经网络模型和 SVM 支持向量机模型进行比较，验证了模型对于处理海量数据所具有更优预测精度。

其次，为了提升模型预测用时性能，本文引入了适用于海量数据高性能处理的分布式并行计算架构，实现基于数据并行训练的 LSTM 短期负荷预测分布式模型。通过创新性地融合 Apache Spark 分布式集群架构和 Google TensorFlow 分布式集群架构，实现同一集群上完成分布式存储形式的数据集装载、模型的分布式并行训练和预测模型部署全过程，大大降低了预测模型训练用时与海量数据集的端对端传输时延。本文通过所搭建的虚拟机集群实验分析了改变集群 worker 节点数量及改变训练并行方式对预测模型用时和精度影响，验证了分布式模型对加速预测模型训练和优化预测模型精度的有效性。

最后，本文从解决复杂工程问题的角度分析了本文模型优缺点和适用范围，提出下一步可行的研究工作，对这一领域未来学术研究具有一定的参考价值和实践意义。

关键词： 短期负荷预测 LSTM(长短期记忆循环神经网络) TensorFlow Apache Spark

Study on Short-Term Load Forecasting Based on High-Performance Mass Data Processing Algorithms

Major: Electrical Engineering and Automation

Student: Yin Sikai Supervisor: Liu Yang

Abstract

The demand on economical operation and accurate dispatch of power system has been increasing during recent years due to the development in smart grid and electricity market reform. The widespread utilization of electricity information collection system and smart power meters has generated mass data of power system. Therefore, the key area of power system operation like short term load forecasting(STLF) need to have the capability of high-performance processing of mass data. In this paper, we take the cutting-edged techniques in power system data science field to propose the short-term load forecasting model with high performance in mass data processing and we utilized real historical load dataset to verify our model. The main contents of this paper are as follows.

Firstly, we proposed the long-short term memory recurrent neural network(LSTM) model in deep learning field for short-term load forecasting. This model was built on Google TensorFlow machine learning framework in Python. After hyper-parameter tuning, the neural network model was utilized to forecast both the aggregated and single-meter load datasets. The accuracy comparison was made between the proposed model, traditional day-ahead load forecasting model, BP neural network and SVM model respectively to verify the performance of this proposed model.

Secondly, we introduced the distributed parallel architecture to realize the forecasting model to achieve high-performance in mass data processing. By combining distributed TensorFlow clusterspec and Apache Spark cluster architecture, we realized the distributed load storage, parallel training of forecasting model and model application on the same cluster. We conducted the experiment to test the performance of proposed model by switching between synchronized and asynchronized training mode and changing the number of worker nodes in cluster. The

performances in these modes were compared by both time consumption and forecast evaluation to show the validity of proposed model.

In the last part, we presented the conclusion of this paper and showed both strengths and weaknesses of the proposed forecasting model. The future work was presented in both dataset and computation hardware aspects. This work is therefore helpful to researches in related fields in the future.

Key words： Short-term load forecasting Long short-term memory recurrent neural network(LSTM) TensorFlow Apache Spark

目 录

1 导 论

1.1 研究背景

近年来，随着我国电力市场改革的深入，电力系统系统运行及其调度的经济性得到了越来越多的重视。由于电力系统的固有特征，要求发电与用电必须时刻保持平衡，因而调度人员通常需要通过合理的负荷预测手段，来制定发电机的启停计划、安排检修任务以及确定备用容量等内容，在未来电力市场中负荷预测还可能与电价激励、负荷响应安排密切相关，因而对这一领域的研究可以促进电源、电网及负荷之间的协调和互动，并达到积极消纳新能源发电，实现电力系统经济调度的目的。据西方学者Bunn and Farmer [1]中对1984年英国电网所做的测算，短期负荷预测每1%的精度提升都能够每年减少超过1000万英镑的调度成本。由此可见，合理精准的负荷预测对于电力系统的经济运行具有重要意义，而负荷预测的误差甚至可能对电力系统的安全稳定与可靠性造成威胁。

当前，学界一般根据所要进行预测的时间尺度，将负荷预测研究划分为超短期、短期、中期、长期等不同类型[2]。在这些不同种类中，短期负荷预测对发电厂出力控制和输电网络分配最为关键，因而长期以来短期负荷预测一直为研究热点方向。由于负荷预测所基于的负荷数据本质上为非平稳连续序列，因此负荷预测问题实质上可以抽象为序列预测问题，可以采用一系列数据科学理论与方法加以处理，此前业界对此也多有论述，所提出的各类经典预测方法对电网的运行做出了巨大的贡献。

然而，随着当前传感器技术和通信技术的发展，电力信息采集系统和智能电表等设备得到了越来越多的推广，从客观角度显著降低了收集电力负荷数据的成本，调度部门对用户的用电信息获取的数量和类型大幅提升，用户侧的数据采集维度复杂，频次增高，所收集的数据规模已经达到TB甚至PB级别[3]，传统负荷预测方面应对不断增长的数据规模性能亟待提升，为大数据技术在短期负荷预测领域的应用提供了客观条件。

在短期负荷预测的诸多算法中，人工神经网络作为一种智能算法长期以来被诸多相关研究所采用。随着近年来数据规模的提升和分布式并行计算的大规模普及，这一算法领域出现了长足的进步和发展，并出现了深度学习这一全新的数据科学领域，利用深度神经网络为主的算法对海量数据加以处理和挖掘，获得海量数据背后所蕴含的规律。当前，深度学习算法已经成功应用于机器翻译、图像识别、文本处理和广告推荐等多个场景[4]，利用人工智能技术提升相关效率。作为物联网系统的一部分，电力系统所产生的数据也同样可以迁移到相关情境，因此利用人工智能与深度学习技术进行电力系统数据处理和分析逐渐得到重视和发展。

1.2 选题意义

精确的负荷预测技术对于电力系统具有重要意义，根据时间尺度的划分负荷预测又包括了长期、中期、短期和超短期等不同种类，其中短期负荷预测一般以小时为尺度，对于电力系统安全与控制、电力市场运营、制定电网调度方案等应用具有突出意义，同时合理的电网调度方案还可以直接或简介促进发电成本的降低和经济效益的提升，从而有效实现能源的资源优化配置进而取得显著的社会效益。

在当前融合物质流、能量流、信息流、业务流和资金流的能源互联网建设过程中，需要大力推动源网荷协调互动，并且积极消纳新能源和实现经济调度，提升能源系统效率，亟需更佳精准的短期负荷预测技术[5]。与此同时，各类信息的融合和数据规模的膨胀对传统短期负荷预测算法构成了严峻挑战，需要着力发展基于先进计算和电力大数据技术的高性能短期负荷预测算法。

在数据科学领域，近年来在分布式计算技术所引发的算力提升基础上取得了长足的发展和进步。人工智能与机器学习领域的分支深度学习取得了在多个应用领域的成功。2016年3月，Google DeepMind公司基于深度学习技术研发的AlphaGo围棋击败人类围棋世界冠军、职业九段选手李世石，一时间让深度学习技术进入了公众视野。2016年11月特斯拉公司发布了Autopilot 2.0自动驾驶系统[6]，宣称具备全自动驾驶功能，该系统背后的数据处理也同样基于深度学习技术。除此以外，深度学习技术也成功应用于广告投放、机器翻译、图像增强、金融决策、预测疾病等诸多领域，从数据科学的崭新角度为这些应用场景提供了全新的解决方案。对于基于时间序列的序列型数据，当前长短期记忆神经网络(LSTM)得到了最为成功的应用。其通过时间递归和权值共享机制，LSTM神经网络兼具处理和预测时间序列数据的能力并解决了循环神经网络长程依赖问题，避免了信息衰减，往往可以作出更为精确的分类和回归。

由于电力系统负荷数据本质上也可类比为非平稳随机序列，因此上述先进计算技术和数据科学算法均可以迁移应用至电力系统领域，并且有望提升在大数据背景下电力系统短期负荷预测的算法性能和预测精度，从而实现前文所述的经济和社会效益，因而值得进行研究和应用。同时作为本科生毕业设计论文，该选题所基于的机器学习框架和分布式计算框架的掌握将能够显著提升本科生的数理分析能力和计算机应用能力，同时研究方向深入整合了电力系统分析、电力系统调度自动化和能量管理系统、自动控制原理、发电厂电气部分和电力系统安控等多个理论课程体系，将综合理论体系应用于电力系统实际应用情境，研究所采用的框架均为该领域内主流通用框架，验证模型所采用的数据集也均采用实际电网负荷数据，因此该研究对于锻炼本科毕业生解决复杂问题的能力、追踪研究前沿热点和掌握终生学习意识具有深远意义。

1.3 研究现状

由于短期负荷预测对电网安全运行的重要作用及其潜在的经济效益和社会效益，这一领域一直是学界研究热点方向。为了充分发掘电网负荷数据的内在规律，往往还会结合不同地区的气候、人口、文化等因此进行综合分析和合理预测。经过几十年的理论研究与发展，短期负荷的预测精度不断提升，相关预测理论也得到不断的发展，国内诸多文献对此多有论述，大致可以划分为传统预测方法和智能预测方法两类：

在短期负荷预测技术发展的早期，所利用的预测模型往往基于调度人员的实际工作经验加以实现。随着电网信息化程度的提高和统计模型的发展，逐渐完成了对电网的数学建模，并采用经典的统计学模型完成电网短期负荷预测工作，形成了传统预测方法。传统预测方法主要包括回归分析法、时间序列法、相似日法、灰色预测法等。这些算法的基本模型原理如下：

对于回归分析算法[7]-[8]，其核心是利用历史负荷数据建立统计回归模型，从而寻找输入量与输出量之间的近似回归方程式，基于最小二乘等准则来确定方程式的参数，并依据回归模型进行预测，其回归方程式的参数与数学形式如下：

$$\mathrm{y(t)} = a_0 + a_1 x_1(t) + \cdots + a_n x_n(t) + \theta(t) \tag{1.1}$$

式中，x(t)为输入量，即历史负荷数据，y(t)为输出量，即负荷数据的预测值，a_i(i=0,1,…,n)为回归系数，$\theta(t)$服从正态分布。

对于时间序列算法[9]，其核心是将负荷数据作为随机时间序列加以分析和预测，通过建立基于实际预测目标序列的随机模型，并估计随机模型中的未知参数，对随机模型加以评估，建立预测表达式的形式得到输出作为负荷预测值，其算法具体可以分为加权时序平均法、加权移动平均法、指数平滑法、自回归移动平均法等等，其中自回归移动平均法（ARIMA）的参数与数学形式如下：

$$\widetilde{x_t} = \theta + \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots + \alpha_p x_{t-p} + \beta_1 x_{t-1} + \beta_2 x_{t-2} + \cdots + \beta_q x_{t-q} \tag{1.2}$$

式(1.2)中，$\widetilde{x_t}$为输出量即t时刻的电力负荷预测值，而x_{t-1}, x_{t-2},…为输入量即t之前时刻的电力负荷数据之间的差分值，$\alpha_i(i = 1,2,\ldots,p)$称为AR系数，即模型时序数据本身的滞后数；$\beta_j(j = 1,2,\ldots,q)$称为MA系数，即模型中预测误差的滞后数。则在模型中的p和q已知的情境下，即可通过被观测系统的时间序列数据获取预测值。

对于相似日法[10]-[11]，主要为了解决负荷数据的非平稳问题。其基本准则是通过构建合理的评价体系获取预测目标的历史相似日，通过相似日的具体历史负荷数据来估算和预测目标日的负荷值。显然，相似日算法的关键在于对相似日的合理选择，文献中往往需要构建日特征向量，以便计算不同日期的相似度情况。其中日特征向量的相似度计算又包括有欧几里得距离法、夹角余弦公式法、因子匹配系数法等等。其中最为精确的因子匹配系数法需要考量包括气象因子、时间因子、星期因子和节假日因子等方面，以提升预测的质量。

对于灰色预测法[12]-[14]，其基于灰色理论，可以利用相对较少的原始负荷数据进行关联分析，从而生成规律性较强指数变化的时间序列，从而解决数学模型的微分方程建模问题，

其本质是对离散数据的累加和求导过程，主要生成数据的方式包括累加生成、累减生成和加权累加生成，通过上述动态建模，可以检验与修改模型精确度。灰色预测通过降低对历史负荷数据量与分布变化规律的依赖，显著降低了负荷数据的统计特征量的要求，因而在传统预测上得到了广泛的应用。

随着人工智能技术的发展，各种新算法理论被越来越多地应用到了短期负荷预测背景中，其中应用较为广泛的智能预测方法包括模糊逻辑法、支持向量机算法、人工神经网络算法以及组合预测方法等。

对于模糊逻辑法[15]-[18]，其基于模糊理论来解决短期负荷数据不平稳和不精确的问题，将短期负荷数据作为一种模糊序列，通过建立合理的隶属度函数来实现非线性映射。模糊预测模型往往需要建立合适的模糊状态集与不同时刻间的模糊状态转移系数，构造模糊概率矩阵并计算预测时刻的模糊向量，最后再基于最大隶属度原则确立预测值。模糊逻辑法最大的优势在于其避免了直接建立传统预测方法的统计模型，可以大大降低预测建模的计算量并提高预测数据的精度。

支持向量机[19]-[21]属于有监督机器学习算法，该算法通过最小化广义误差的上界来达到最大推广能力。为了解决数据的非线性可分问题，支持向量机算法通过引入核函数$K(x, y)$，可以将输入的历史负荷数据向量x映射到更高维度特征空间中，从而实现在该高维空间中构建可分超平面的方法，通过机器学习训练最小化测试误差可以实现短期负荷预测效果，因而该算法相较于时间序列法等传统算法而言在非线形高维数据模型处理方面具有明显的优势。

人工神经网络算法[22]-[30]从上世纪80年代起逐渐成为人工智能领域的研究热点，其本质是模拟生物体大脑的工作过程，通过建立合适的网络结构，利用历史负荷数据作为输入，并根据合理的训练算法让网络自动确定其中的网络权值和参数，从而根据精度要求实现对网络结构的参数确定，进而实现对非线性高维度数据模型的建立，并将模型用于未来短期负荷的预测。由于人工神经网络的上述训练需要较为大量的数据，并且训练花费的时间往往受限于具体的机器算力，因此在近年来随着分布式计算和硬件算力的提升也得到了明显的发展并在深度学习领域取得了全新的突破，因而本文所主要研究的负荷预测算法也将主要基于这一机理，在后文进行详细的介绍。

除了上述的标准经典预测方法以及智能预测方法外，中外很多研究文献还创造性地根据电力系统短期负荷预测应用背景对原始算法进行了合理的改进和优化，或结合多种不同预测算法各自的优点适用范围进行合理组合形成组合预测方法。例如在文献[31]中，作者通过季节因素结合自回归积分滑动平均模型(ARIMA)，文献[32]提出了基于ARIMA的转移方程模型，文献[33]将ARIMA模型与专家系统相结合用于预测，文献[34]中更是将非高斯随机过程引入ARIMA模型中。文献[35]最早开始将支持向量机算法引入短期负荷预测领域，而文献[36]将支持向量机与遗传算法组合形成预测模型，文献[37]中更是将支持向量机组合相似日法进行组合预测。对于神经网络预测，文献[38]中基于传统人工神经网络引入了径向基函数RBF神

经网络进行短期负荷预测，而文献[39]中将多层感知机与混合Levenberg－Marquardt算法相结合，结合自适应组合遗传算法进行误差反向传播训练网络；文献[40]和[41]中还将遗传算法引入人工神经网络的训练过程，形成组合模型优化模型预测结果。基于对上述文献的阅读和算法思想的理解，本文将着重结合人工神经网络中的深度学习模型提出对短期负荷的预测模型，以期获得更好的预测效果。

1.4 电力系统短期负荷预测工作流程

对于电网短期负荷预测工作，其实施具有严谨的工作流程和研究方法，从数据产生的源头即数据采集开始，一直到数据的各种处理，模型的建立和预测结果的输出与预测误差的分析，每个环节都需要严谨科学的理论指导，才能确保数据的质量，进而决定预测结果的精确性和价值。这一短期负荷预测工作的基本流程如下图1所示：

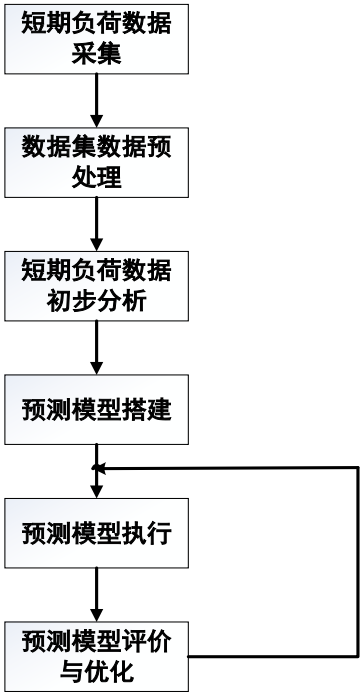

图 1 短期负荷预测工作流程图

根据以上流程图，短期负荷预测工作的具体内容应包括以下6个部分:

（1）短期负荷数据采集

正如前文所述，由于当前用电信息采集系统和智能电表的逐步普及，以及传感器技术、仪器仪表制造技术的进步，对短期负荷数据的采集工作一般可以通过调度中心由SCADA系统获取，通过网络连接将所需数据收集并保存至主机硬盘。数据采集工作必须确保细致专业，以避免收集数据的质量与完整性对后续预测步骤产生不利影响。

（2）数据集数据的预处理

对于所采集的短期负荷数据，由于无法保证采集传感器的老化、损坏以及网络通信造成的数据干扰问题，因此对采集的原始数据集往往需要先进行数据清理和预处理工作，主

要包括填补缺失值、调整异常值等，具体采用的方法在相关手册中多有论述，本文不再赘述。

（3）短期负荷数据初步分析

在完成数据集中数据的预处理工作后，一般为了确定后续预测模型中的一些参数和初始值，会先对短期负荷数据进行初步分析。由于负荷数据往往具有周期性，包括日周期、周周期、季周期和年周期等，可以对其数据平稳性和特征进行初步分析，倘若有条件的话还可以进行负荷分类等工作，以便于后续预测步骤的进行。

（4）预测模型搭建

在完成负荷数据的初步分析后即可根据实际需要搭建相应的预测模型，选择合适的预测算法往往需要在计算时间和预测精度之间做出取舍，应根据负荷数据实际特点和工作实际需求进行合理选择。对于本文中的人工神经网络模型，在此步骤应确定网络结构和网络超参数调整，以确保预测模型可以以较高精度和收敛速度执行。

（5）预测模型执行

在此环节，通过输入现有负荷数据集，执行预测模型并输出预测结果进行合理保存，以便后续进行模型评估和参数优化工作。

（6）预测模型评价与优化

在预测模型评价和优化部分，往往可以通过建立多个预测模型并在一定准则下计算并比较模型性能，并做好模型优化的工作。对于本文中的短期负荷预测模型，后文所采用的预测精度评估标准主要有以下两种：

（1） 平均绝对误差（mae）

$$\mathrm{mae}=\frac{1}{N}\sum_{i=1}^{N}|E_i|=\frac{1}{N}\sum_{i=1}^{N}|y_i-\tilde{y}_i| \tag{1.3}$$

（2） 均方误差（mse）

$$\mathrm{mse}=\frac{1}{N}\sum_{i=1}^{N}|E_i|=\frac{1}{N}\sum_{i=1}^{N}(y_i-\tilde{y}_i)^2 \tag{1.4}$$

上式中，其中mae为平均绝对误差，mse为均方误差，N为测试集样本总数量，y_i为第i个测试样本真实负荷值，$\tilde{y}_i$为模型预测负荷值。

1.5 论文思路与结构

本文作为电气工程及其自动化专业本科毕业设计，深入全面地研究了基于深度学习领域中长短期记忆循环神经网络（LSTM）的电力系统短期负荷预测模型，对实际电力负荷数据进行合理预处理后，通过Google TensorFlow机器学习平台编程实现上述短期负荷预测模型，将LSTM预测模型的预测精度与传统日前负荷预测方法、BP神经网络和SVM支持向量机模型等进行对比，验证在海量数据处理背景下模型的预测精度。之后本文通过融合Apache Spark分布式集群架构和分布式TensorFlow机器学习集群架构，提出了短期负荷预测模型的

分布式并行计算框架，并实现LSTM预测模型的分布式并行计算，能够在同一分布式集群上完成海量数据集存储、预测模型并行训练和预测模型部署应用工作，从而验证模型在海量数据处理背景下的应用价值。本文所提出的这一结合深度学习算法和分布式高性能计算框架的短期负荷预测模型在实际生产中的应用具有坚实基础，对这一领域的相关研究具有一定的指导意义。本文总体技术路线的思路与结构图如下：

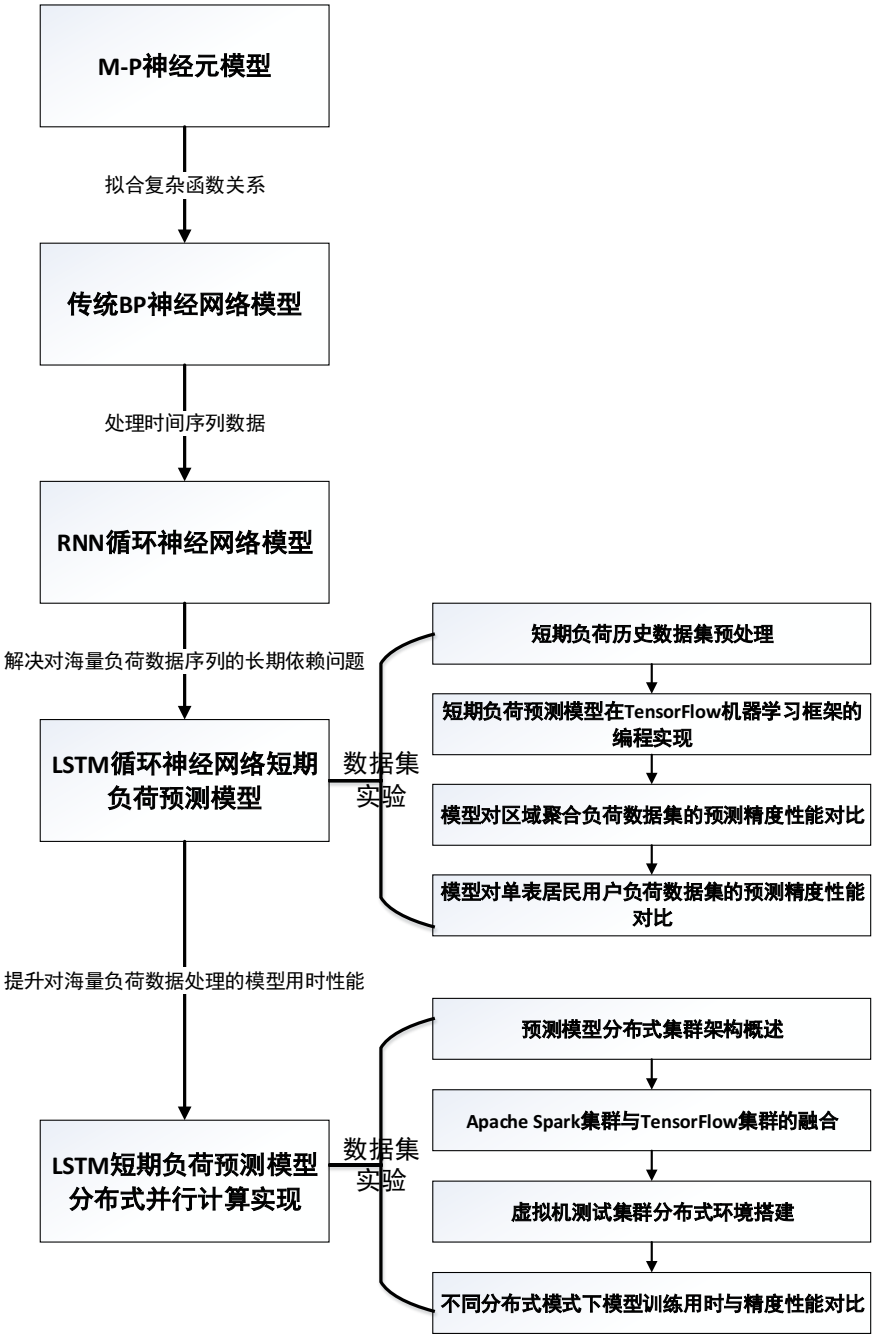

图 2 论文全文结构图

2 人工神经网络模型理论

2.1 传统人工神经网络模型

人工神经网络最早起源于1943年，心理学家Warren S.McCunoch和数学家Walter H.Pitts共同发表论文《A Logical Calculus of the Ideas Immanent in Nervous Activity》[42]，将生物神经系统归纳为“M-P神经元模型”，提出了基于这一生物神经网络的逻辑运算模型，开创了人工神经网络研究的先河。之后感知机、自适应非线性元件等人工神经网络基础模型在60年代被陆续提出。1986年，David E.Rumelhart, Geoffrey E.Hinton和Ronald J.Williams共同发表论文《Learning Representations by Back-propagating Errors》[43],利用实验方式验证和发展了误差反向传播（BP）算法对更新人工神经网络权重和促进人工神经网络训练的有效性，解决了非线性分类问题，从而为多层人工神经网络的训练和实际应用奠定了可行基础。2010年后，随着GPU硬件的普及和分布式计算框架的应用，对人工神经网络的研究逐渐成为热点，并且逐渐开拓出深度学习这一全新的发展领域。2012年，来自多伦多大学的Geoffrey Hinton等人利用名为AlexNet的深度卷积神经网络架构在ImageNet计算机视觉图像分类竞赛中取得性能的极大突破，引领了对神经网络的研究热潮。2015年，在ImageNet比赛中微软亚洲研究院的工作人员利用全新的深度残差网络架构首次超越了经过训练的人类对这一数据集的分类成绩，再次印证了神经网络研究的巨大潜力和广阔前景。2016年来自Google DeepMind公司开发的AlphaGo围棋基于深度强化学习训练，在比赛中战胜了人类围棋世界冠军李世石九段[44]，举世瞩目。如今，人工神经网络技术已经发展出包括感知机、多层前馈神经网络、径向基函数神经网络、卷积神经网络、循环神经网络等不同的神经网络结构，网络权值的训练方法也由过去的反向传播算法逐渐扩展到更多方法，性能愈发强大的人工神经网络正在图像和物体识别、电子游戏、语音生成和识别、金融预测与辅助决策、自动驾驶控制等领域获得越来越多的成功应用，带领人类走向基于大数据的人工智能时代。

从上世纪九十年代伊始，人工神经网络理论作为一种解决非线性高维度数据建模的有效方法，被学者逐步应用到电力系统的短期负荷预测工作中。由于短期负荷数据往往具有较高的随机性和非线性，因而各种传统预测模型的算法参数不易于设置，各种预测模型对短期负荷的预测精度一直有提升的空间。随着电力大数据当前的不断发展，这一海量数据处理和应用场景也正在得到愈发关键的体现，将能够为未来电力系统的效率提升作出长远的贡献。

2.1.1 神经元结构

人工神经网络的基础结构所模仿的是人脑结构，任何人工神经网络的基础单位都是由一个个神经元所构成的，这类神经元也就是上文提及的M-P神经元模型，其结构如下图所示：

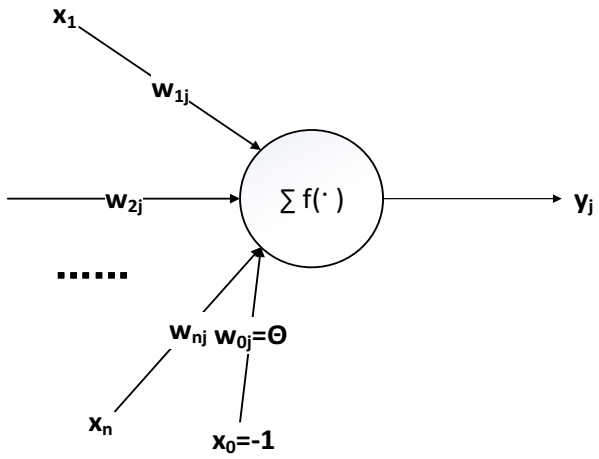

图 3 M-P 神经元模型结构图

对于单个神经元而言，其本质即为输入与输出之间的逻辑映射，其映射关系可以由下式所表达：

$$y_j = f\left(\sum_{i=1}^{n} w_{ij} \cdot x_i - \theta_j\right) \tag{2.1}$$

其中j为该神经元的表示，则y_j即为该神经元的映射输出，x_i（i=1,2,…,n）即为神经元的多个输入，w_{ij}为各输入所对应的连接权值，θ_j为阈值，一般设定为常量。f一般称为激活函数，很大程度上决定了神经元输入与输出之间的映射关系，一般是非线性函数，可以向神经网络的映射中引入非线性因素。根据函数性质的不同，激活函数又可以分为阈值型、分段线性型、连续型等用于处理。对于每个神经元，只有当输入值的总和超过阈值θ_j时，该神经元才会被激活。若将该阈值θ_j也作为神经元的一个输入$x_0 = -1$的权重w_{0j}，则单个神经元的映射关系为：

$$y_j = f\left(\sum_{i=0}^{n} w_{ij} \cdot x_i\right) \tag{2.2}$$

2.1.2 BP 神经网络结构

由于单个神经元的映射效果十分有限，因此要真正解决电力系统短期负荷预测等随机非平稳序列的建模和处理必须依靠神经元组合成多元多层的复杂神经网络。在本文概述部分已经提到，在人工神经网络发展过程中误差反向传播算法的提出和应用对这一领域的进步起到了极大的推动作用，因为BP算法通过误差的反向传播能够用于修正网络连接权值，从而为多层神经网络的学习提供了可行的途径。基于这一算法的广泛影响，后世将利用BP算法训练的多层前馈神经网络称为BP神经网络或多层感知机[45]。其基本结构如下图所示：

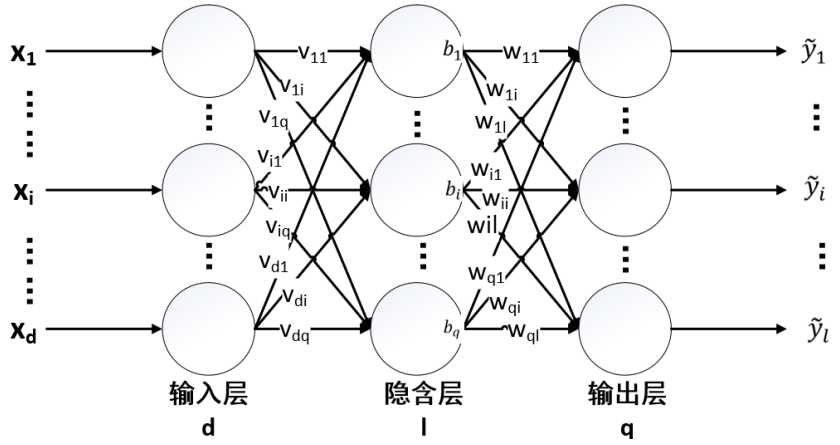

图 4 BP 多层前馈神经网络结构图

为简化讨论，以上图4三层构成的BP神经网络为例，对于输入神经网络的数据，假设数据集具有m个样本，每个样本记为(x,y)，若每个样本的输入有d个属性值，输出有l维实数值向量，则训练用数据集$D=\{(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)\}$，$x_i\in R^d, y_i\in R^l$。若神经网络输入层神经元数量为$d$，隐含层神经元数量为$q$，输出层神经元数量为$l$，则可以对训练样本在神经网络中的运算加以表示。此外定义隐含层第h个神经元的阈值为γ_h，输出层第j个神经元的阈值为θ_j,输入层第i个神经元与隐含层第h个神经元之间的连接权值为v_{ih},隐含层第h个神经元与输出层第j个神经元之间的连接权值为w_{hj}，则隐含层第h个神经元总输入值为$\alpha_h=\sum_{i=1}^{d}v_{ih}x_i$，记隐含层第$h$个神经元的输出为$b_h$，则输出层第$j$个神经元总输入值为$\beta_j=\sum_{h=1}^{q}w_{hj}b_h$，各层激活函数均极为$f$，则对于样本数据集中的训练样本$(x_k,y_k)$，神经网络的前向传播后的输出结果即为：

$$\tilde{y}_j^k=\left(\tilde{y}_1^k,\tilde{y}_2^k,\dots,\tilde{y}_l^k\right) \tag{2.3}$$

其中，

$$\tilde{y}_j^k=f\left(\beta_j-\theta_j\right) \tag{2.4}$$

网络在(x_k,y_k)样本的均方误差为

$$L_k=\frac{1}{2}\sum_{j=1}^{l}\left(\tilde{y}_j^k-y_j^k\right)^2 \tag{2.5}$$

2.1.3 BP 神经网络训练方法

在理解BP神经网络的前馈结构后，对BP网络的学习规则即误差反向传播算法进行概述。对于上述BP神经网络，对其任一样本(x_k,y_k)，由该神经网络计算得到的误差函数为

$$L_k=\frac{1}{2}\sum_{j=1}^{l}\left(\tilde{y}_j^k-y_j^k\right)^2 \tag{2.6}$$

而BP神经网络的学习过程误差反向传播（BP算法）就是通过样本训练调整权重系数，使得上述总误差函数L以迭代的形式向误差减小最快的方向收敛，直到满足要求的误差范围内的过程。对于上图中的BP神经网络，其需要训练的网络参数有输入层和隐含层之间的

$d\times q$个连接权值，隐含层和输出层之间的$q\times l$个连接权值，q个隐含层神经元的阈值以及l个输出层神经元的阈值。在迭代的每一轮中，BP算法都采用广义的感知机学习规则对参数进行更新估计，上述这些中任意参数θ 的更新估计式都可以表达成

$$\theta \leftarrow \theta + \Delta\theta \tag{2.7}$$

对于单个样本误差反向传播算法而言，对于图4中的BP多层前馈神经网络，以隐层到输出层神经元连接权值参数为例，设其每轮迭代的修正量为Δw_{hj}，由于BP算法基于梯度下降策略，即以优化目标函数的负梯度方向以学习速率η的单位值进行调整，则对于上文得到的误差函数L_i而言，每轮迭代的修正量为

$$\Delta w_{hj} = -\eta \frac{\partial L_k}{\partial \mathrm{w}_{hj}} \tag{2.8}$$

由于在该BP神经网络中，隐层到输出层神经元连接权值参数w_{hj}先影响到第j个输出层神经元的输入量β_j，然后再影响到该神经元的输出值$\tilde{\mathrm{y}}_j^k$，进而影响样本误差函数L_i，因此再计算梯度下降时，根据微积分中的链式法则可得如下表达式：

$$\frac{\partial L_k}{\partial \mathrm{w}_{hj}} = \frac{\partial L_k}{\partial \tilde{\mathrm{y}}_j^k} \cdot \frac{\partial \tilde{\mathrm{y}}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial \mathrm{w}_{hj}} \tag{2.9}$$

根据$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$，则

$$\frac{\partial \beta_j}{\partial \mathrm{w}_{hj}} = b_h \tag{2.10}$$

另根据激活函数的不同，对链式法则中的部分步骤的运算可以简化，则简化起见在通式中不妨设

$$g_j = -\frac{\partial L_i}{\partial \tilde{\mathrm{y}}_j^k} \cdot \frac{\partial \tilde{\mathrm{y}}_j^k}{\partial \beta_j} = -\left(\tilde{\mathrm{y}}_j^k - y_j^k\right) f'\left(\beta_j - \theta_j\right) \tag{2.11}$$

$$e_h = -\frac{\partial L_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} = -\sum_{j=1}^{l} \frac{\partial L_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h) = \sum_{j=1}^{l} \mathrm{w}_{hj} g_j f'(\alpha_h - \gamma_h) \tag{2.12}$$

那么上文中所有的网络参数在BP算法中的更新公式即可简记为

$$\Delta w_{hj} = \eta g_j b_h \tag{2.13}$$

$$\Delta \theta_j = -\eta g_j \tag{2.14}$$

$$\Delta v_{ih} = \eta \mathrm{e}_h x_i \tag{2.15}$$

$$\Delta \gamma_h = -\eta e_h \tag{2.16}$$

除了上文所推导的基于单样本的标准BP算法以外，实际在海量负荷数据处理背景下应用更多的是累积误差BP算法，其更新规则为训练样本数据集上的累积误差最小化原则。累积误差BP算法只在完全读取训练集$\mathrm{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$一遍后才执行一次参数更新过程。对于上述含有m个样本的训练集，其训练集总误差函数为：

$$L=\sum_{k=1}^{m}L_k=\frac{1}{2}\sum_{k=1}^{m}\sum_{j=1}^{l}\left(\tilde{y}_j^k-y_j^k\right)^2 \tag{2.17}$$

通过这种累积误差的BP算法，参数更新频率更慢一些。综上所述，对BP神经网络训练的误差反向传播算法总结如下：

表 1 误差反向传播算法

输入	训练集$D=\{(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)\}$ 学习速率η

过程	按随机初始化方案进行所有网络待训练参数的初始化
	while 训练终止条件未达成 do
	for each $(x_k,y_k)\in D$
	得到当前样本输出值$\tilde{y}_j^k$
	计算输出层神经元的梯度项g_j
	计算隐含层神经元的梯度项e_h
	计算网络更新参数$\Delta w_{hj},\Delta v_{ih},\Delta\theta_j$和$\Delta\gamma_h$
	更新网络待训练参数w_{hj}，v_{ih}，θ_j和γ_h
	end for
	end while

输出	网络参数确定的BP多层前馈神经网络

2.2 LSTM 循环神经网络模型

2.2.1 循环神经网络概述

与传统的BP神经网络相比，循环神经网络在结构和功能上均有一定的不同。对于循环神经网络，其结构内定义了数据存储机制，在不同时刻按时序输入数据序列，从而可以更好地处理序列数据问题。循环神经网络并不仅仅对单一时刻的输入进行输出和预测，其可以通过存储机制将本时刻部分隐含层神经元的输出结果予以保留，并且与下一时刻的网络输入共同作用于下一时刻的网络神经元，使得每一时刻的网络输出都受到该时刻之前所有历史序列的影响。

从结构上而言，循环神经网络的神经元结构存在有环，使得历史输入序列可以再次影响当前时刻神经元的状态，实际上实现了同隐含层的不同神经元之间的信息传递[46]。循环神经网络的基本神经元结构如下图所示，其中U为输入层到隐含层之间神经元的连接权值，V为隐含层内神经元的连接权值，W为隐含层到输出层之间的神经元连接权值，{U,V,W}称为网络权值参数集，具有在时间上权值共享的机制。由于神经元环的存在，实际上可以在时序上将该环展开，从而将循环神经网络神经元展开为下图形态：

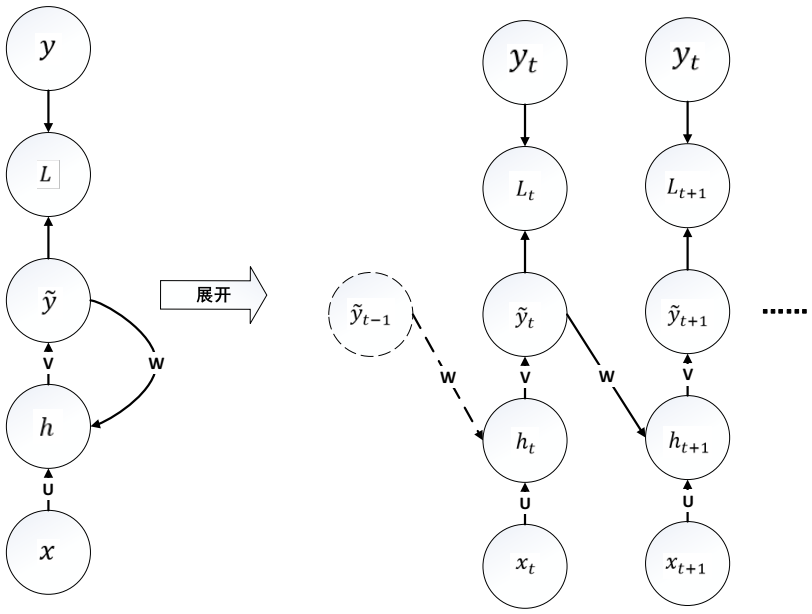

图 5 循环神经网络神经元及其展开结构图

通过这一RNN神经元展开结构可知，循环神经网络可以实现在不同的时刻传入不同的输入数据，并且结合上一时刻神经元保存的状态共同作用于当前时刻的输出结果。

在实际应用中，为了完成对随机性和非线性较强的电力负荷数据预测，往往需要不只一层的RNN隐含层，因而构成含有多个隐含层的循环神经网络结构，所输入的数据值经过多层网络映射可以达到更好的参数表达效果，并充分发挥深度学习算法的优势。

在循环神经网络中，根据序列数据的性质不同可以将网络部署成如下几种不同的形式。第一种基本形式称为单对多，即循环神经网络将单个输入序列转化为多个时刻的输出序列，实现输入与输出之间的映射；第二种基本形式称为多对单，即有多个时刻进行输入序列而只在最后时刻输出单个序列，有利于整合历史序列数据；第三种基本形式称为有时延多对多，即输入层和输出层均有多个时刻的序列，且网络的输入与输出序列之间有一定的时间延迟，可以较好的通过网络训练完成各类序列数据的预测工作；第四种基本形式称为无时延多对多，即输入层和输出层均有多个时刻的序列但序列的输入输出时刻不存在时间延迟，这一结构有利于让循环神经网络完成各种即时性处理，例如图像识别和自然语言识别等。上述几种循环神经网络基本结构形式如下图所示：

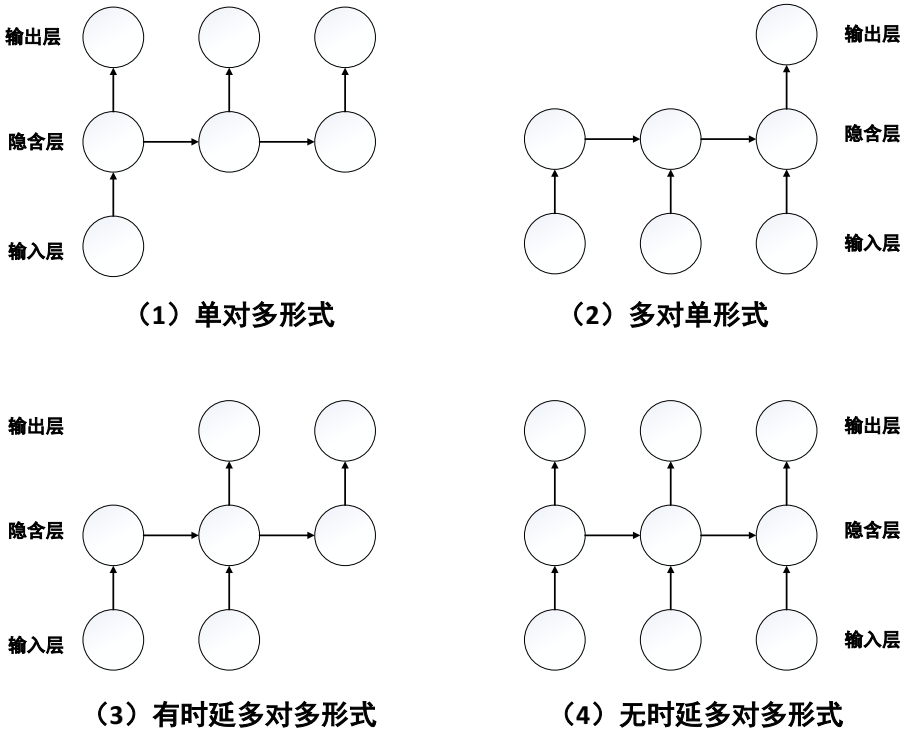

图 6 循环神经网络的四种基本应用形式

此外，循环神经网络模型具有网络权值共享的特征，即随着时刻的推移，其网络权值参数集$\{U, V, W\}$不随时刻发生改变。通过这一网络权值集合权值共享机制，降低了网络训练的开销，网络所需要学习更新的参数对于每个神经元只有该权值集合中的权值量，网络中所不同的仅仅是输入与输出序列。

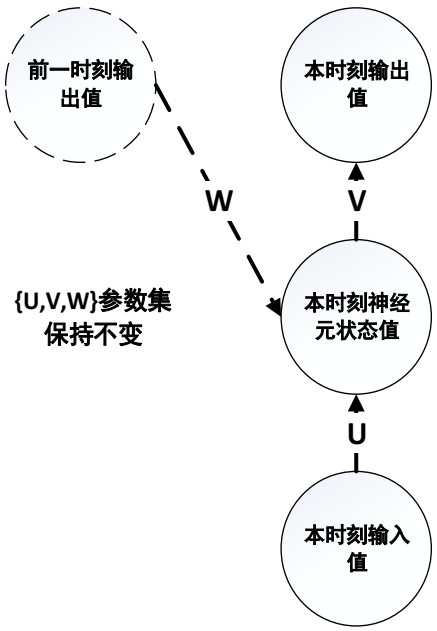

图 7 循环神经网络的权值共享机制

基于循环神经网络的上述特性，可以分别推导循环神经网络的前向传播传播和模型训练算法。对于前向传播过程，首先可以初始化网络状态简记为h_o，则对于起始时刻$\mathrm{t} = 1$直到结束时刻$\mathrm{t} = \mathrm{T}$，每时刻的循环神经网络输出为：

$$\alpha_t = Wh_{t-1} + Ux_t + \mathrm{b} \tag{2.18}$$

$$h_t = f(\alpha_t) \tag{2.19}$$

$$\tilde{y}_t = Vh_t + c \tag{2.20}$$

其中，x_t为该时刻的网络输入，h_t为该时刻的隐含层激活值，f为隐含层激活函数，$\tilde{y}_t$为该时刻的网络输出，b和c分别为输入层和隐含层的偏置向量，$\{\mathrm{U},\mathrm{V},\mathrm{W}\}$为网络权值参数集。

对于循环神经网络的训练过程，可以发现在网络权值集合$\{\mathrm{U},\mathrm{V},\mathrm{W}\}$中，隐含层和输出间的神经元连接权值W更新可以直接通过损失函数loss对相应的权值W求导实现，即适用于前文介绍的BP神经网络的误差反向传播算法，而权值U,V实际上与时间有关。为了能够训练循环神经网络，必须采用基于时间的误差反向传播算法（BPTT）进行训练。对于展开的RNN网络中的每个节点N，若定义其损失函数为L，则在训练过程中需要对梯度值即为$\nabla_N L$进行迭代计算，其具体梯度值计算需要基于该节点之后时刻节点的的梯度值进行。因此不妨从最后终止时刻该节点的最终损失函数开始进行梯度值求取，则有

$$\frac{\partial L}{\partial L_t} = 1 \tag{2.21}$$

则对于参数值V的梯度值，可以计算为

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial V} \tag{2.22}$$

$$\frac{\partial L_t}{\partial V} = \frac{\partial L_t}{\partial \tilde{y}_t} \cdot \frac{\partial \tilde{y}_t}{\partial \mathrm{V}} \tag{2.23}$$

对于参数值W和U，以参数W为例，则需要进行计算为

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W} \tag{2.24}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \tilde{y}_t} \cdot \frac{\partial \tilde{y}_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W} \tag{2.25}$$

由于此处h_t本身为该时刻的隐含层激活值，则是与W相关的值，因此基于时间进行计算，可以得到其值为

$$\frac{\partial L_t}{\partial W} = \sum_{\tau=0}^{t} \frac{\partial L_t}{\partial \tilde{y}_t} \cdot \frac{\partial \tilde{y}_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial h_\tau} \cdot \frac{\partial h_\tau}{\partial W} \tag{2.26}$$

由于误差反向传播的梯度值计算过程中需要对全部时刻进行链式法则求导，因此基于该梯度值进行的模型训练与参数更新称为基于时间的误差反向传播算法（BPTT）。循环神经网络这一记忆时序的特征使得其处理时间序列数据的性能理论上可以达到远高于传统人工神经网络的水平，并在序列数据处理领域得到了大范围的应用。从数据本质上而言，短期电力负荷数据同样具有严格的时序特征，其特定时刻的负荷数据值往往与该时刻之前的负荷数据有密切关联，因此十分适合采用循环神经网络进行处理。

在理想条件下，含有多隐含层的循环神经网络可以处理任意时间长度的序列型数据并产生有效预测，然而实际上循环神经网络存在“长期依赖”问题，即在前文中基于时间进行误差反向传播时，对于上式（2.26）可以链式法则表示为

$$\frac{\partial h_t}{\partial h_\tau}=\frac{\partial h_t}{\partial h_{t-1}}\cdot\frac{\partial h_{t-1}}{\partial h_{t-2}}\cdot\ldots\cdot\frac{\partial h_{\tau+1}}{\partial h_\tau} \tag{2.27}$$

当该链式法则中相邻的两项h_t之间的梯度值小于1时，上式即会在几个时刻的h_t值范围内接近于0，这意味着较远时刻的h_t值已经无法对本时刻的输出值产生影响，即意味着出现了循环神经网络记忆部分的信息衰减，由于该衰减是由于梯度趋于零的计算现象导致的，因此又称之为“梯度消失”现象。在这一情况下，循环神经网络对较近几个时刻的输入值产生反应，因此这一现象又称为循环神经网络RNN的长期依赖问题。

为了解决这种标准循环神经网络RNN模型的长期依赖问题，研究人员提出了新的循环神经网络结构，通过改变网络神经元内部结构确保网络可以持续保存和记忆较远时刻的输入值和隐含层状态，确保对序列数据的预测效果，这种网络结构被称为长短期记忆循环神经网络LSTM，也是本文中进行电力系统短期负荷预测的核心算法模型。

2.2.2 LSTM 神经网络结构

为了解决标准的循环神经网络RNN的长期依赖问题，在其基础上发展出了LSTM循环神经网络模型，其本质仍属于RNN循环神经网络的一种，因此同样适用于上文对RNN的基本前馈传播和网络训练内容。LSTM神经网络最早由Hochreater和Schmidhuber于1997年提出[46]，主要目的即是解决在基于时间的误差反向传播（BPTT）算法中的梯度消失和梯度爆炸问题，引入了CEC单元(constant error carrousel)。在2001年，Felix Gers更进一步改进了LSTM网络的结构[47]，增加了遗忘门（forget gate）和peephole结构，从而形成了目前主流的LSTM算法。目前这一网络结构已经成功被谷歌、百度、微软、苹果、亚马逊等诸多知名公司所采用，在语音识别、代码生成、机器翻译等等领域有效解决了长期依赖导致的信息衰减问题。对于海量数据背景下的电力负荷预测，由于数据存在较强的随机性，因此显然也需要解决长期依赖问题，因此本文所提出的高性能海量数据预测算法基于LSTM神经网络模型。

对于LSTM网络，其不仅具有标准循环神经网络RNN中的神经元链状结构，且具有更为复杂的隐含层神经元结构。LSTM神经网络结构如下图所示：

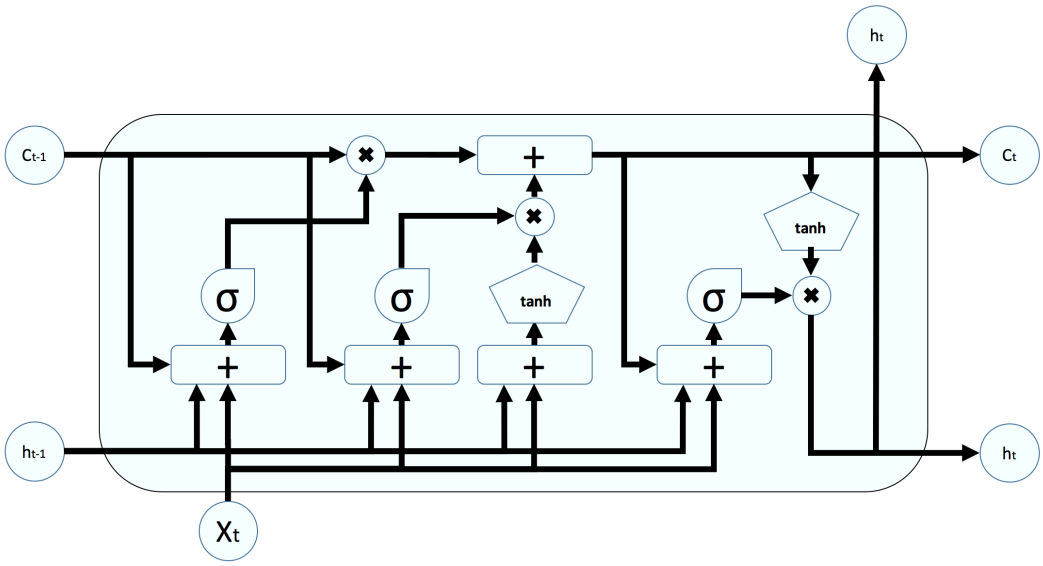

图 8 LSTM 循环神经网络结构图

在如上图所示的LSTM神经网络神经元中，实际上包含有四层神经网络，每层隐含层中实际包含了一个自连接存储单元以及三个乘法单元门，分别为输入门、输出门和遗忘门，从而为LSTM单元分别提供读、写和删除操作。

对于LSTM神经网络，其解决长期依赖问题的主要结构即为乘法门。连接多个隐含层单元的横线包括两个逐点操作，具有和RNN相同的链状结构效果以储存单元值。而在该横线上的两个逐点操作添加值主要由下方的三个乘法门决定，由乘法门的参数可以向该链状结构内有效的添加或删除信息，乘法门的参数决定了信息是否可以通过。综上所述，对于每个LSTM单元在相邻时刻间输入与输出序列之间的运算关系如下：

$$\mathrm{f}_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.28}$$

$$\mathrm{i}_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.29}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.30}$$

$$C_t = \mathrm{f}_t * C_{t-1} + \mathrm{i}_t \tilde{C}_t \tag{2.31}$$

$$\mathrm{o}_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.32}$$

$$h_t = \mathrm{o}_t * \tanh(C_t) \tag{2.33}$$

上式中，f_t为LSTM单元在前一时刻单元值C_{t-1}在经过遗忘门后保留状态量的比率，0表示完全舍弃前一时刻单元值信息，1表示完全保留。W_f即为遗忘门的参数设置为0-1之间，b_f为遗忘门的偏置量。i_t为LSTM单元在此时刻伊始由上一时刻的输出和此时刻的输入共同作用于单元状态的输入值，W_i即为输入门的参数设置为0-1之间，b_i为输入门的偏置量。$\tilde{C}_t$为输入门产生的另一部分候选值向量，将和i_t作乘法后产生最终的输入量。因此在当前时刻最终的单元状态值即为前一单元状态信息值经过遗忘门和输入门后的值$C_t = f_t * C_{t-1} + \mathrm{i}_t \tilde{C}_t$。最后$\mathrm{o}_t$为由前一时刻的输出序列和当前时刻的输入序列所共同产生的当前时刻输出值中间变量，在该中间变量与当前时刻单元的激活函数值tanh (C_t)后，共同决定当前时刻的LSTM网络输出为h_t。

2.2.3 LSTM 神经网络激活函数

在LSTM循环神经网络中，激活函数主要用于给神经网络引入非线性因素，LSTM网络结构中主要有两种不同的常见激活函数，分别为σ Sigmoid激活函数和tanh激活函数。

Sigmoid激活函数被称为S型生长曲线，其接受各种输入后的映射输出范围为(0,1)，因此可以将实际数值压缩让数据容易聚拢。对于特别大的负数Sigmoid函数映射为0，即位完全未激活状态，而特别大的正数则由Sigmoid函数映射为1，即为完全激活状态。Sigmoid激活函数表达式为：

$$\mathrm{sigmoid(x)} = \frac{1}{1+e^{-x}} \tag{2.34}$$

tanh激活函数实际上是一种双曲正切函数，与Sigmoid函数相比它能够将输入值映射到[-1,1]的输出范围内，即很大的负数映射为-1，很大的正数映射为1，0的映射恰仍为0，符合对称性特点。tanh激活函数表达式为：

$$\tanh(\mathrm{x}) = \frac{\sinh x}{coshx} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.35}$$

2.2.4 网络训练 Dropout 机制

Dropout是神经网络训练过程中的一种正则化技术，最早在文献[48]中被引入和应用。在搭建LSTM网络预测模型的过程中，有必要防止神经网络训练过拟合，因此本文在构建LSTM网络的过程中通过设计合理的dropout比例来构建模型。Dropout机制所设定的参数为p，其可以在每次迭代前对网络中每个隐含层的神经元以p概率舍去，1-p的概率保留，从而每次隐藏1-p隐含层神经元，则能过随着迭代的进行有效避免过拟合。

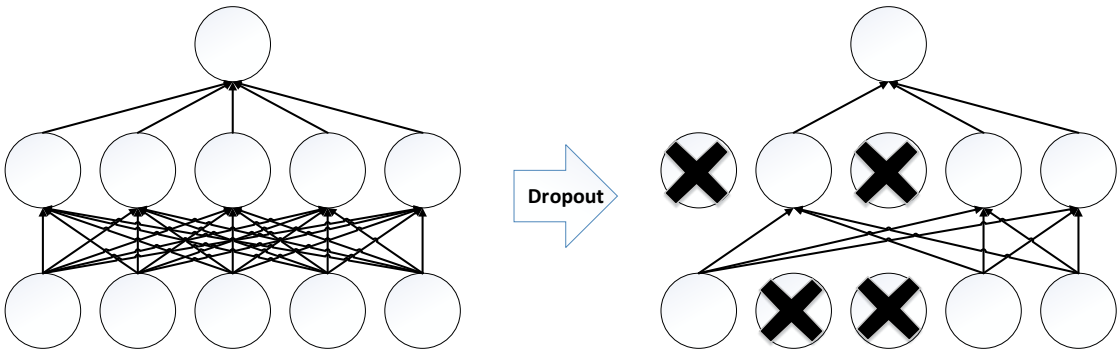

图 9 深度神经网络训练的 Dropout 机制

2.3 本章小结

在本章部分，从M-P神经元的结构开始，对传统的BP神经网络以及用于处理序列数据的RNN循环神经网络以及LSTM神经网络的结构、优势、前馈预测以及训练方法进行了概述，着重对比了传统BP神经网络与LSTM循环神经网络之间的差异，便于后文我们在搭建电力系统短期负荷预测模型时利用这两类不同的神经网络模型进行预测结果的性能对比和分析。实际上在本章中介绍的各类神经元与神经网络结构之间有清晰的逻辑发展关系，即从最开

始提出的M-P神经元开始，为了表达更复杂的抽象关系有了BP多层前馈神经网络。针对处理序列数据的输入输出特点，对神经网络模型赋予了记忆和存储机制，因此有了结构带自环的循环神经网络RNN模型，而该模型在处理长序列数据时时常出现梯度消失而引起长期依赖问题，因此最终提出了LSTM循环神经网络模型。这一个个神经网络模型的发展脉络和承接关系时十分清晰的，也是本文后文在选择模型进行短期电力负荷时的考量，因此本章可以作为后文模型应用的理论基础和分析。在从BP多层前馈神经网络到循环神经网络的发展过程中，其模型训练中的梯度计算方法也从传统的BP误差反向传播算法进一步改良为BPTT基于时间的误差反向传播算法，对后文的模型训练也具有重要的理论指导意义。

此外，本章部分还引入了神经网络模型在网络结构之外的一些重要定义，包括LSTM神经网络的激活函数、循环神经网络的权值共享机制和训练深度神经网络过程中常用的Dropout机制等等，在下文将神经网络具体迁移到短期负荷预测领域中搭建预测模型时，也需要利用以上理论进行模型搭建和模型训练。

3 基于 LSTM 神经网络的电力系统短期负荷预测模型

3.1 TensorFlow 机器学习框架概述

本文所采用的所有短期负荷预测模型的实验均采用Google TensorFlow机器学习计算框架。作为当前最主流和最受欢迎的机器学习框架，TensorFlow由美国谷歌公司在2015年11月9日正式开源[49]。TensorFlow实质上是一种采用数据流图（data flow graphs）用于数值计算的开源软件库，具有高度灵活的架构，可以支持在多种硬件平台上展开计算和实现模型（如台式计算机中的一个或多个CPU（或GPU），以及服务器或手机等移动设备），其高度的灵活性、真正的可移植性、自动求取导数、多编程语言支持、性能管理优化及科研与生产产品联系在同类机器学习平台中具有明显的优势。

对于谷歌而言，其处于核心地位的排序系统RankBrain是基于TensorFlow框架写成，在谷歌公司的核心搜索业务中占据重要地位，并且在内部的广告、电商、地图、翻译、视频和语音处理等多个业务领域都得到了广泛的应用，谷歌的工程师们一直在积极使用这一框架提供用户直接在用的产品和服务，并且对推动TensorFlow开源社区的活跃度做出了巨大贡献，从客观上促进了TensorFlow的发展。此外，学术界和工业界也对这一机器学习平台十分重视，据最新统计目前每月发表的机器学习领域相关论文中有14.3%均采用TensorFlow框架进行算法验证和数据实验。因其代码规范、文档齐全、社区支持性强，因此本文最终选择使用这一主流的机器学习框架进行算法实验。

对于TensorFlow机器学习架构，其数值计算主要基于数据流图来实现。每个数据流图实质上都是由数值结点（nodes）和线（edges）组成的有向图，用来表示相关的数值计算。数据节点一般用来表示施加的数学操作，也可表示数据输入（feed in）的起点和输出（push out）的终点，或是读取/写入永久数据变量（persistent variable）的终点。线则表示数据节点之间的输入/输出关系，这些线可以传输规模可动态调整的多维数据数组，这些数组也被称为即张量（tensor）。对于每个TensorFlow程序都会创建相应的数据流图，当输入张量被准备完毕，TensorFlow就会启动一个会话（session），将节点分配到各种计算硬件设备完成模型运算，并得到相应的模型参数和结果。下图10即为一个包括张量（tensor）、变量（variable）和操作（operation）三种图节点构成的计算流图示例：

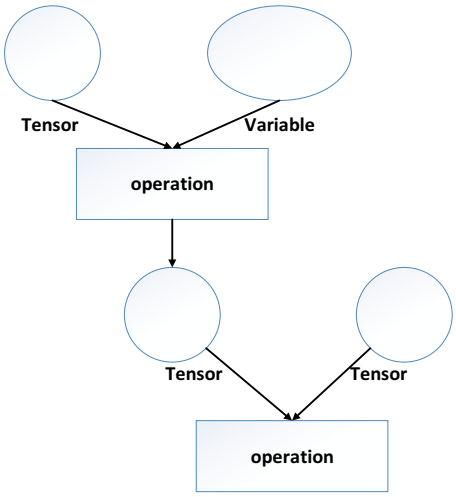

图 10 Google TensorFlow 计算流图示例

对于本文中所研究的短期电力负荷预测问题，要实现在TensorFlow框架内的模型搭建和预测大致分可为模型数据流图搭建、数据读取、模型训练、模型测试四个部分，在首先进行数据集的数据预处理工作后，需要在这一框架上将研究所使用的LSTM网络模型通过数据流图的形式加以实现。对于数据流图中的输入部分，由于本文中的电力负荷数据需要从CSV文件形式读取，因此必须采用输入变量作为数据流图的输入节点，在数据流图中使用placeholder占位符进行表示，在读取文件后模型实际运行训练时再由feed操作将实际输入负荷数据投喂到模型中。对于数据流图中的模型部分，TensorFlow在nn.rnn_cell库中提供了标准LSTM神经网络层，事实上LSTMCell相当于前文所述的LSTM隐含层，在其内部包含有遗忘门、输入门、输出门等部分，可根据需要设定初始值和隐含层节点个数等网络参数，我们将在后文实验部分进行超参数寻优后予以确定。此外数据流图中还要保存网络权值部分变量，可将网络权值均设为Variable类型便于模型运行过程中进行训练和更新。最后在数据流图中设定寻优参数，包括优化器、优化准则及学习速率等超参数，即可完成数据流图部分的工作。搭建模型数据流图后，最后通过调用tf.session创建会话进行模型的运行，经过训练和迭代后在测试集上进行测试，即可根据所设置的一定的损失函数和误差作为准则来评判模型在短期电力负荷数据集上的性能和表现。

3.2 实验数据准备

3.2.1 数据来源

本文中用于短期负荷预测的数据分为两部分，即区域聚合历史负荷数据集和单表居民用户历史负荷数据集，分别用于从聚合和单用户尺度对后文的预测模型进行评估验证以及性能优化。其中聚合负荷值数据集取自欧洲互联电网（ENTSO-E）的信息发布平台ENTSO-E Transparency Platform（https://transparency.entsoe.eu/load-domain/r2/totalLoadR2/show）中瑞士自2015年1月1日到2017年4月30日共849天的实际负荷数据（Actual Total Load）和ENTSO-E所发布的日前预测数据（Day-ahead Total Load Forecast）。导出的聚合负荷数据所有数据值

均以1小时为时间间隔，共计20376条负荷数据记录，符合本文所需的短期负荷预测数据要求。所有数据均用CSV文件格式导出，并且通过python pandas库进行数据初步处理，将上述所有日期数据进行合并为一个整体CSV文件，以备后文分析。经过数据初步分析发现，该聚合数据发布较为权威且不存在缺失值和异常值，因此无需进行多余的数据预处理工作，可直接用于后文分析。

本文的单表用户负荷数据取自欧洲互联电网（ENTSO-E）爱尔兰区域单表负荷数据，所有负荷数据经过脱敏处理后共包含1000用户单表，用户电表编号即为1000—1999，每电表采集共536天数据，天数编号即为195—730，导出负荷数据值均以0.5小时为时间间隔，每表共计25728条符合数据记录，符合本文所需的短期负荷预测数据要求。所有数据均用Txt文件格式导出，并且通过python pandas库进行初步处理后以单表为单位分割导出为1000个CSV文件，已备后文分析。经过数据初步分析发现，该单表用户负荷数据存在有数据缺失和异常问题，大多数单表文件实际记录数不足，需要后文进行专门的数据预处理工作，以保证原始数据质量。上述两类历史负荷数据集数据格式如下表2、表3所示，由于负荷数量较多，只输出一天的部分为例：

表 2 聚合负荷历史数据集数据（部分）

Time (CET)	Day-ahead Total Load Forecast [MW]	Actual Total Load [MW]
01.09.2016 00:00 - 01.09.2016 01:00	6248	6063
01.09.2016 01:00 - 01.09.2016 02:00	6213	5684
01.09.2016 02:00 - 01.09.2016 03:00	6095	6107
01.09.2016 03:00 - 01.09.2016 04:00	6008	6113
01.09.2016 04:00 - 01.09.2016 05:00	5727	6097
01.09.2016 05:00 - 01.09.2016 06:00	5459	5845
01.09.2016 06:00 - 01.09.2016 07:00	6622	6049
01.09.2016 07:00 - 01.09.2016 08:00	7546	6191
01.09.2016 08:00 - 01.09.2016 09:00	7605	6982
01.09.2016 09:00 - 01.09.2016 10:00	7836	7411

表 3 单表居民用户负荷历史数据集数据（部分）

Meter_index	Time	Value
1984	19501	0.256
1984	19502	0.271
1984	19503	0.327
1984	19504	0.219
1984	19505	0.162
1984	19506	0.157
1984	19507	0.043
1984	19508	0.152
1984	19509	0.078
1984	19510	0.071
1984	19511	0.163
1984	19512	0.043
1984	19513	0.114
1984	19514	0.125
1984	19515	0.043
1984	19516	0.149
1984	19517	1.638
1984	19518	0.05
1984	19519	0.808
1984	19520	0.079
1984	19521	0.07

表 2（续）		
01.09.2016 10:00 - 01.09.2016 11:00	7999	7556
01.09.2016 11:00 - 01.09.2016 12:00	7736	7806
01.09.2016 12:00 - 01.09.2016 13:00	7826	7996
01.09.2016 13:00 - 01.09.2016 14:00	8008	7488
01.09.2016 14:00 - 01.09.2016 15:00	7924	7574
01.09.2016 15:00 - 01.09.2016 16:00	7860	7522
01.09.2016 16:00 - 01.09.2016 17:00	7528	7457
01.09.2016 17:00 - 01.09.2016 18:00	7154	7324
01.09.2016 18:00 - 01.09.2016 19:00	7236	7105
01.09.2016 19:00 - 01.09.2016 20:00	7077	7044
01.09.2016 20:00 - 01.09.2016 21:00	7111	6874
01.09.2016 21:00 - 01.09.2016 22:00	6771	6869
01.09.2016 22:00 - 01.09.2016 23:00	6714	6676
01.09.2016 23:00 - 02.09.2016 00:00	6054	6358

表 3（续）		
1984	19522	0.157
1984	19523	0.518
1984	19524	1.719
1984	19525	2.406
1984	19526	1.724
1984	19527	0.231
1984	19528	0.578
1984	19529	0.241
1984	19530	0.105
1984	19531	0.17
1984	19532	0.149
1984	19533	0.128
1984	19534	0.661
1984	19535	0.955
1984	19536	0.474
1984	19537	0.329
1984	19538	0.309
1984	19539	0.423
1984	19540	0.427
1984	19541	0.167
1984	19542	0.279
1984	19543	0.255
1984	19544	0.16
1984	19545	0.357
1984	19546	0.455
1984	19547	0.396
1984	19548	0.452

3.2.2 数据预处理

对于短期电力负荷预测任务而言，历史负荷数据的数据质量对预测模型的建立和精度提升起着至关重要的作用。对于从智能电表及用电信息采集系统所收集的负荷数据，其数据质量可能会因人工操作失误、计量设备老化或损坏、通信链路中断或受到干扰等因素而受到影响，产生无法满足精度要求的“脏数据”，为了确保后文预测模型的精度及预测有效性，必须首先对数据集进行预处理，识别并排除“脏数据”的影响。

对于本文中的单表用户负荷数据，首先针对传感器表计漏测的数据进行缺失值填补。由于实际采集数据的缺失值并不多，低于5%左右的统计阈值，且考虑到对于30min的采集间隔，其负荷用电量与前后2个横向负荷点统计差异很小，因此本文中采用前后数据均值填充的方法进行缺失值处理。假设所缺失的数据值为第i天的第j点负荷数据，则对该负荷数据填补所采用的处理公式如下：

$$x_{i,j} = \frac{1}{2}\left(x_{i,j-1} + x_{i,j+1}\right) \tag{3.1}$$

之后对负荷数据的异常值进行处理。在本文中可以利用横向比较法进行脏数据的识别和处理，即通过负荷数据的周期性特征设置样本统计指标和阈值，对于不满足的负荷数据进行修正。对于本文单用户负荷数据，基于用户负荷的日周期性特征，首先计算历史同时刻的负荷数据均值与方差，并设置阈值加以判断，对于处理后发现的异常值数据通过前后负荷点数据进行修正，计算均值及方差公式如下：

$$\overline{x_{i,J}} = \frac{1}{536}\sum_{i=1}^{536} x_{i,j}\,, j = 1,2,\ldots,48 \tag{3.2}$$

$$\sigma_{\mathrm{i}}^{2} = \frac{1}{536}\sum_{i=1}^{536}\left(x_{i,j} - \overline{x_{i,J}}\right)^{2} \tag{3.3}$$

之后根据统计学的3σ原理进行异常值判断，将阈值ε设为1.5代入以下公式：

$$\left|x_{i,j} - \overline{x_{i,J}}\right| > 3\sigma_i\varepsilon \tag{3.4}$$

对于发现的异常值数据$x_{i,j}^{*}$，利用以下公式进行修正，权重$\alpha + \beta = 1$：

$$x_{i,j}^{*} = \frac{1}{2}\alpha\left(x_{i,j-1} + x_{i,j+1}\right) + \beta\overline{x_{i,J}} \tag{3.5}$$

经过上述基本预处理，解决了样本数据集中的缺失值和异常值问题。接下来需要对数据进行规范化处理，让样本数据实现归一化，在不损失数据特征的基础上将数据统一映射到[0,1]区间上。通过数据归一化可以提升预测模型的收敛速度，并且提升预测模型的精度。目前数据标准化方法有多种，归结起来可以分为直线型方法(如极值法、标准差法)、折线型方法(如三折线法)、曲线型方法(如半正态性分布)。在本文中对数据进行归一化的方法为zero-score归一化，即利用原始数据的均值和标准差进行规划化处理，所得的归一化数据均值为0，方差为1。所采用的标准化公式如下：

$$x'_{i,j} = \frac{x_{i,j} - \mu}{\sigma^{2}} \tag{3.6}$$

公式中μ为数据集整体均值，σ^{2}为数据集整体方差，而$x'_{i,j}$即为处理后的标准化数据。

经过上述预处理处理步骤后所得的区域聚合负荷数据集和单表居民负荷数据集中的历史负荷 随时间变化的归一化数据曲线依次如下：

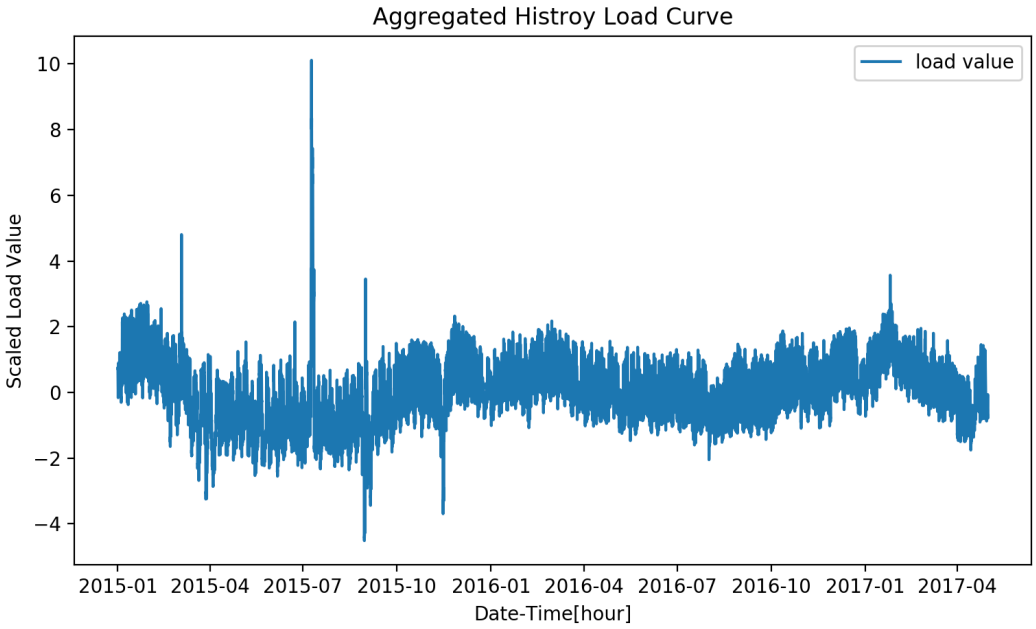

图 11 聚合负荷数据集历史负荷归一化数据曲线

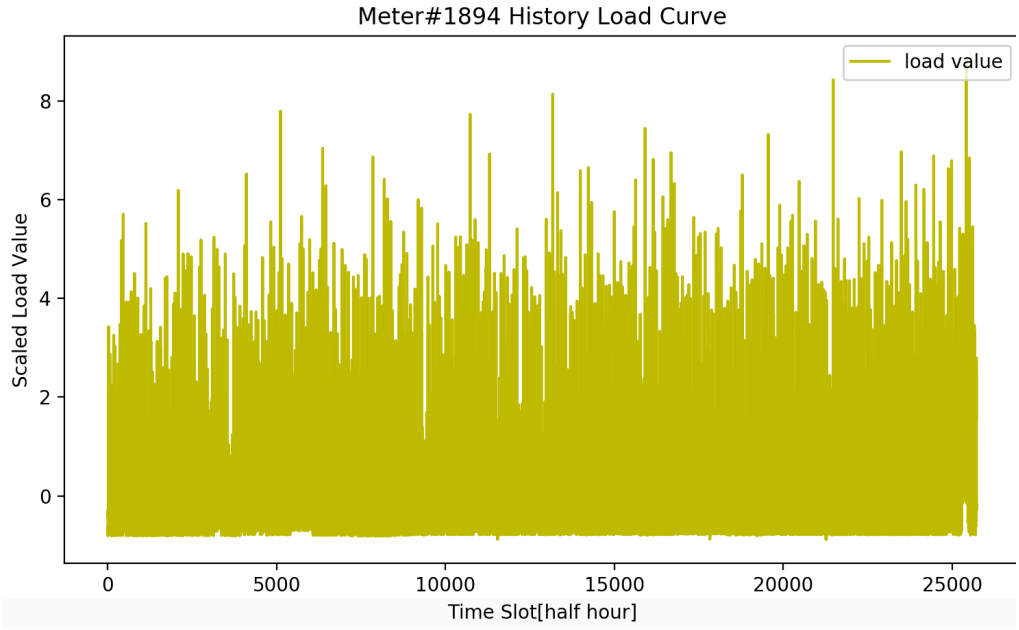

图 12 1894 号单表居民用户负荷数据集历史负荷归一化数据曲线

对聚合负荷数据集数据曲线和单表居民用户负荷数据集曲线加以对比可知，两种负荷的数据曲线形态具有明显的差别，两种负荷均具备明显的日周期特征，并且在归一化后的曲线可以看出聚合负荷曲线的数值平稳性显著优于单表居民用户负荷，居民用户负荷时间序列的随机性和波动性更强，也是一类更难以作出精准预测的负荷数据。因此对于本文所搭建的LSTM预测模型将在这两类不同的数据特征的负荷数据集上分别加以性能测试，并且与传统短期负荷预测方法的性能加以对比，来确定本文所提出的模型的有效性和预测性能情况。

3.3 LSTM 预测模型的网络超参数调优

如前文所述，LSTM神经网络模型中由误差反向传播（BP）和基于时间的误差反向传播算法（BPTT）训练所更新的权重主要为网络权值集合$\{u, v, w\}$，而其余网络结构设置和关键超参数必须通过一定的方法配置和调整。因此对于本文所要实现的短期负荷预测任务，我们将基于部分样本在避免过拟合的前提下运用贪婪算法进行配置调参。对于LSTM神经网络模型算法而言，其中最关键的几个超参数调整依次如下：

3.3.1 网络 LSTM 隐含层数量

正如前文所述，由于神经网络隐含层实际完成了非线性映射，因此随着隐含层层数的增加，神经网络理论上可以以更加高度抽象的形式学习所输入的训练集内容，这也是深度学习相较于经典神经网络优势所在。对于LSTM网络而言，时间传播和网络深度都可能对算法的预测性能产生影响。因此此处我们在TensorFlow上编程时充分利用了贪婪算法思想，通过前文预处理过的聚合负荷数据确定神经网络层数。首先根据经验假设固定所有层数的神经元均为50个，并且利用dropout机制避免过拟合问题，设定dropout层的参数为0.1，即以概率p=0.1舍弃神经元并让其它神经元以概率q=1-p=0.9予以保留。设定batch规模为24，训练轮数epoch最大值为30，并设置Early Stopping对训练轮数提前截断，避免出现过拟合现象。完成Early Stopping截断后保存模型，利用测试集的平均误差mae和mse值判断层数优劣。

表 4 模型 LSTM 层数与测试集上模型性能表

LSTM 层数	平均绝对误差 mae	均方误差 mse
1	0.339	0.179
2	0.301	0.145
3	0.363	0.205
4	0.359	0.206

根据模型在测试集上的表现可知，在其余网络超参数均保持不变的前提下，LSTM网络并非隐含层越多越好，并且随着LSTM层数的增多可以明显发现训练过程中要达到Early Stopping停止准则所需要的训练轮数epoch增加，并且每轮训练的耗时也会大大增长，对计算机算力有更高的要求。因此根据本文中所需处理的数据实际情况，将选择采用两层LSTM神经网络的结构进行训练。

3.3.2 网络每层神经元数量

由于本文中神经网络结构均为LSTM输入层后接若干LSTM隐含层，经过Dropout层后连接Dense层到输出，因此在确定了神经网络深度为4层（2层LSTM层，1层Dropout层和1层Dense输出层）后，本文将进一步确定LSTM网络每层神经元的最优数量组合。根据网络调参经验，将第1层LSTM网络的可能神经元数量设为10，50，100，150，第2层LSTM网络可能神经元数量设置为0，10，50，100，150，共产生$4\times5=20$种不同的神经网络结构，对这20种神经网络结构分别进行训练，仍设定dropout层的参数为0.1，即以概率p=0.1舍弃神经元并让其它神经元以概率q=1-p=0.9予以保留。设定batch规模为24，训练轮数epoch最大值为

30，并设置Early Stopping对训练轮数提前截断，避免出现过拟合现象。训练完成的模型在测试集的平均误差mae和mse尺度下预测性能表现如下：

表 5 模型各层神经元数量与测试集上模型性能表

性能排名	第一层 LSTM 神经元数量	第二层 LSTM 神经元数量	平均绝对误差 mae	均方误差 mse
1	150	50	0.311	0.154
2	10	0	0.312	0.157
3	150	0	0.315	0.156
4	100	150	0.319	0.162
5	10	10	0.322	0.168
6	100	50	0.323	0.165
7	150	10	0.323	0.164
8	50	10	0.327	0.169
9	150	100	0.329	0.172
10	10	100	0.330	0.171
11	100	10	0.335	0.179
12	50	150	0.337	0.177
13	10	50	0.340	0.182
14	100	0	0.342	0.183
15	100	100	0.342	0.184
16	50	100	0.346	0.187
17	50	0	0.349	0.188
18	150	150	0.360	0.199
19	50	50	0.371	0.213
20	10	150	0.416	0.271

根据模型在测试集上的表现可知，在其余网络超参数均保持不变的前提下，两层LSTM神经网络的神经元个数分别设置为150和50时网络预测精度性能表现最优，因此对于本文的预测模型将依此加以设置。

3.3.3 模型优化器选择

对于LSTM深度神经网络训练而言，其在每个batch数据输入进神经网络获得输出，用于求取loss损失函数，并根据优化器的设置情况通过误差反向传播及基于时间的误差反向传播算法对网络权值进行更新，根据不同的优化模型，TensorFlow给出了不同的优化器模型，其性能对于不同的数据集性质和稀疏性具有不同的优化表现。因此在确定网络结构后本文通过默认设置的优化器首先确定模型的最优优化方式，所给出的备选优化方式包括随机梯度下降法（Stochastic Gradient Descent, SGD）, Adam法，Adagrad法和RMSprop法，并将前文所确定的LSTM模型网络结构运用这些优化器分别在30轮训练轮数下进行模型训练，得到交测试集上的平均误差，得到结果如下表所示：

表 6 模型优化器与测试集上模型性能表

	训练轮数	平均绝对误差 mae	均方误差 mse
SGD	30	0.336	0.179
RMSprop	30	0.419	0.268
Adam	30	0.308	0.149
Adagrad	30	0.377	0.192

根据模型在测试集上的表现可知，在其余网络超参数均保持不变的前提下，选用Adam优化器的模型预测性能表现最优，实际上其同时获得了 Adagrad 和 RMSprop 算法的优点，也是当下优化器中较为理想的选择，因此在后文的讨论中，将全部设置为Adam优化器进行模型训练。

3.3.4 模型学习速率选择

在LSTM网络权值更新的过程中，学习速率（learning rate）是对模型训练结果具有很大影响的一个超参数。由于以模型参数为自变量，损失loss函数为因变量所绘制的loss曲面图往往是非凸的曲面，可能存在多个局部最优点或鞍点，因此学习速率对于让模型在合理时间内收敛到满意精度具有关键意义。一般而言，过小的学习速率（learning rate）可能会导致模型收敛速度过慢，而过大的学习速率（learning rate）可能会导致模型跳出全局最优点甚至使模型无法收敛。为了选取合适的学习速率，在本文短期负荷模型中我们根据前文已确定的超参数，为模型选择相同的优化方法为前文所确定的Adam法,将随机梯度下降的学习速率按照10倍步长原则分别设置为lr=0.0001,0.001,0.01,0.1，在取消Early Stopping情况下分别训练30轮，绘制训练过程中样本集和测试集上的mse损失函数随训练轮数变化曲线如下图所示：

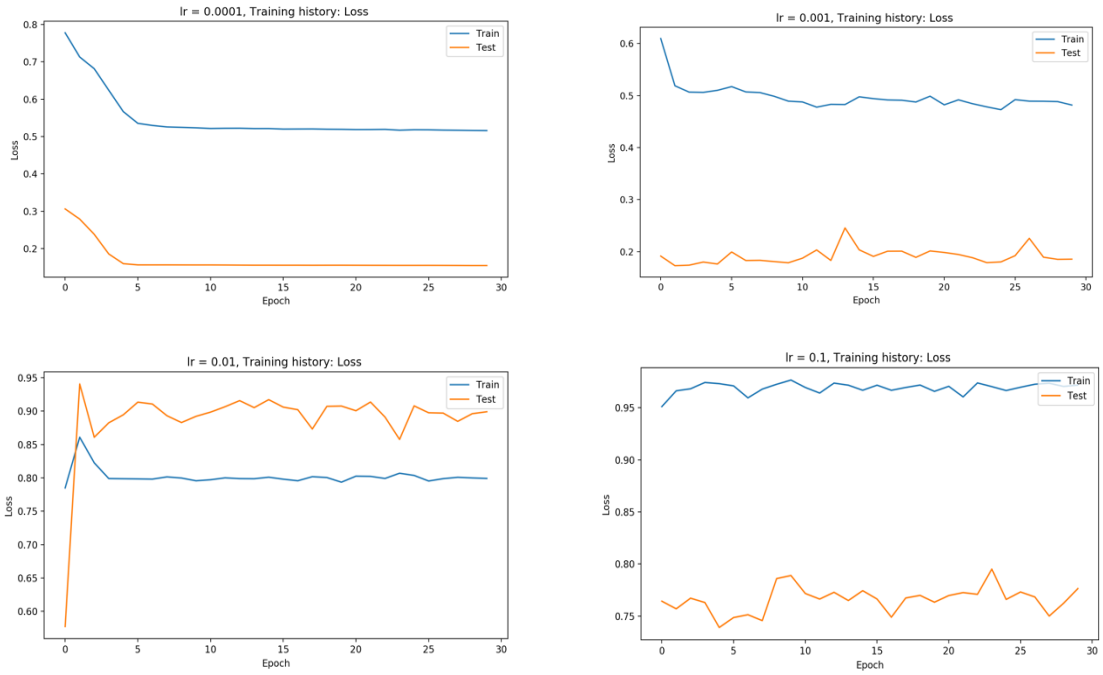

图 13 网络学习速率 lr=0.0001, 0.001, 0.01, 0.1 对应 mse 损失函数随训练轮数变化曲线

根据mse损失函数随训练轮数变化曲线，可以发现当学习速率lr设置为0.1或0.01时损失函数曲线出现振荡现象，无法随着训练轮数的增加保持收敛；而学习速率lr为0.001或0.0001时模型均能够很好的使损失函数收敛，同时lr=0.0001时只需约4轮训练即可收敛到稳定的损失函数，而lr=0.001时训练需要大约20轮，可见对于Adam优化方法由于其具备自适应学习速率加速收敛，因此将学习参数设置较小可以优化训练，与预期结果一致。因此最终为模型确定训练速率lr=0.0001。

综上所述，在进行模型的网络超参数调优后，最终确定的神经网络结构如下图所示：

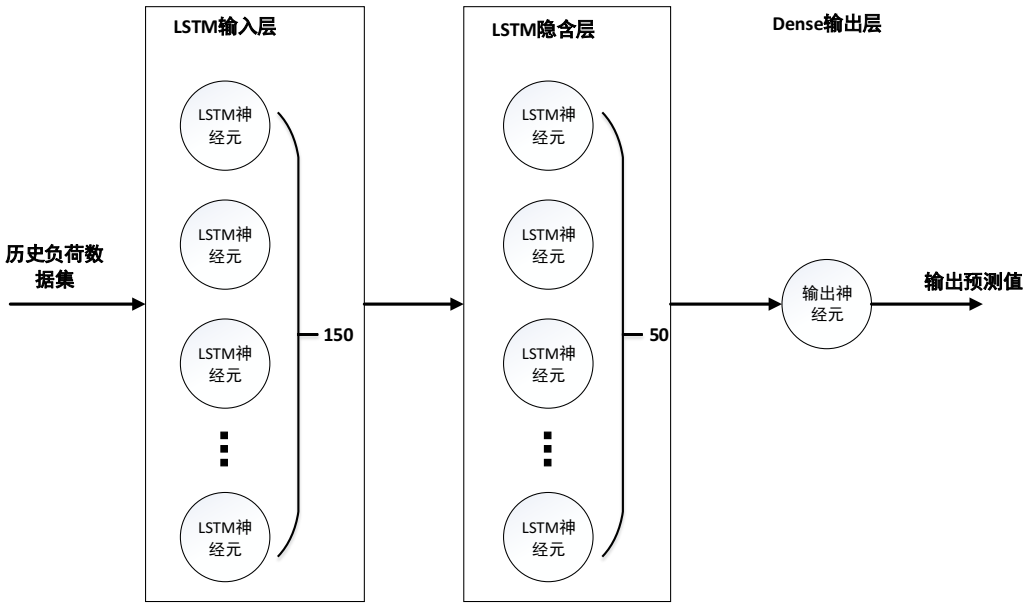

图 14 确定的 LSTM 预测模型网络结构

与此对应的确定的神经网络超参数设置情况表如下表 7 所示：

表 7 神经网络预测模型超参数设置表

网络参数	设置情况
LSTM 层数	2
每层神经元数量	150+50
优化器	Adam
学习速率	0.0001

3.4 LSTM 模型电力系统短期负荷预测精度性能对比

3.4.1 聚合负荷数据集预测结果与模型性能对比

经过前文所述步骤，我们确定了对解决短期电力负荷预测问题LSTM神经网络模型的最优超参数设定，则可以对数据集进行合理划分为样本集与测试集，分别对经过前文预处理的聚合短期负荷和单表用户负荷进行预测。

对于聚合负荷，根据前文3.2.1节可知数据集中已经包含有历史实际负荷数据及通过传统预测方法预测的日前负荷预测数据，因此为了考察LSTM模型的预测精度性能，对聚合负荷数据按每时刻将日前负荷预测值作为输入量，而将实际历史负荷数据值作为标签（label）

以供神经网络学习和完成参数调整。则每个输入神经网络的第i时刻的负荷数据集可以表示为$(日前负荷数据x_i, 实际负荷数据y_i)$。

对于训练集和测试集的划分采用了日期划分的方式，将数据集中2017年2月1日前的数据集作为训练集，而将2017年2月1日后的数据集作为测试集。此外在模型训练中采用交叉验证，划分比例采用“二八原则”，即使用80%的数据进行训练集而采用20%的数据作为验证集。综上所述，原始数据中共有14592条训练数据，3648条验证数据和2136条测试数据。对上述数据输入上文所确定的LSTM网络模型进行预测，在测试数据上所得结果如下图所示：

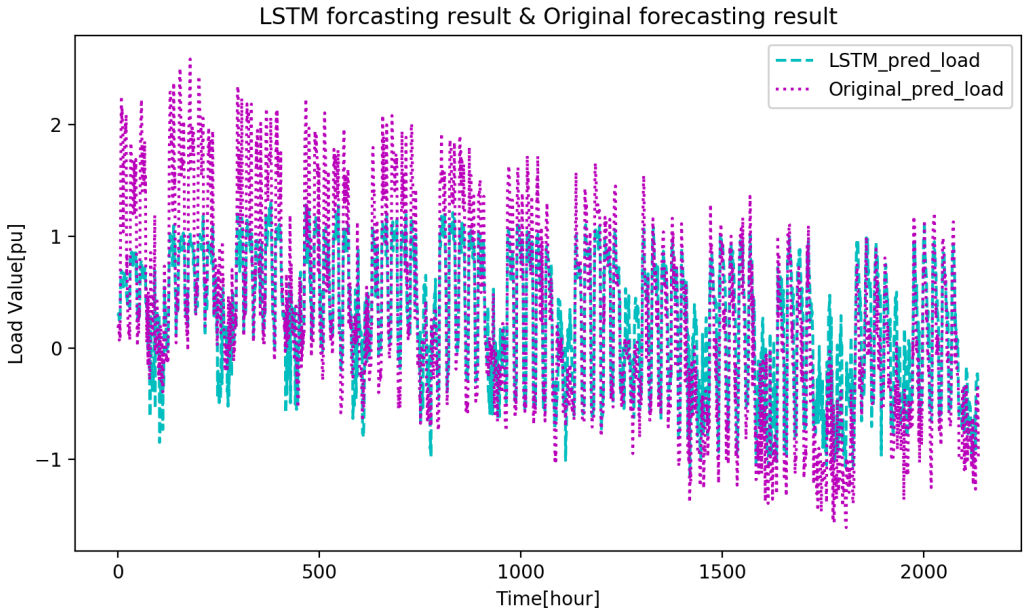

图 15 聚合负荷测试集 LSTM 预测与原日前预测负荷曲线对比图

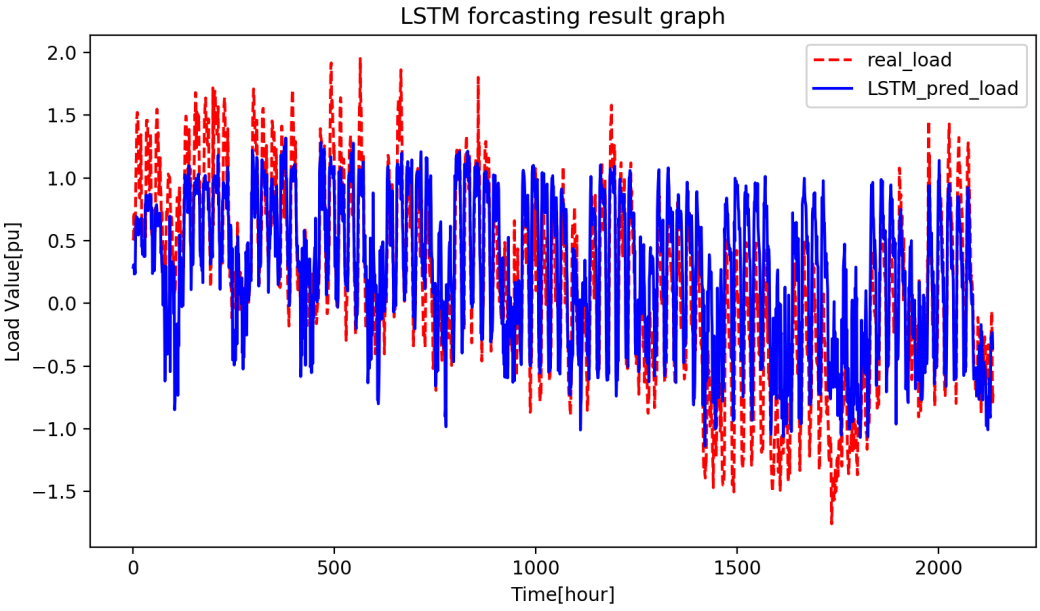

图 16 聚合负荷测试集 LSTM 预测与真实负荷曲线对比图

将测试集上的LSTM预测结果数据同原日前负荷预测数据的误差值加以计算，对比二者的误差大小，可以确定两种算法的预测性能情况：

表 8 聚合负荷数据集上 LSTM 模型与原日前负荷预测模型预测性能对比结果

	LSTM 模型预测模型	原日前预测模型	误差降低百分比
平均绝对误差 mae	0.33697	0.36833	8.51%
均方误差 mse	0.19337	0.21175	8.68%

由上表结果可知，LSTM预测模型在平均绝对误差mae和均方误差mse尺度下均在测试集上比之前的传统预测方法所得日前预测负荷数据精度有所提升，预测误差降低的百分比尺度分别为8.51%（mae）和8.68%（mse），从而证明了通过LSTM预测模型可以对现有短期负荷预测方法加以预测精度改进。

3.4.2 单表居民负荷数据集预测结果与模型性能对比

对于单表用户负荷数据，负荷集中只有历史负荷数据，因此必须将数据集合理划分为训练集、验证集和测试集。此处我们选取10%的数据作为测试集，在样本集中采用“二八原则”进行交叉验证划分，即20%的样本用作验证集，而80%的样本用作训练集。综上对于每个单表用户负荷数据，我们有18284条训练数据，4572条验证数据和2539条测试数据。由于前文已经确定了较好的LSTM神经网络模型的超参数，因此此处继续调用该LSTM网络模型，仍为两层LSTM神经网络，由Dense层输出预测结果，并仍采用Adam优化器进行参数寻优。由于单表居民负荷用户数据集中只有历史负荷数据，因此对网络进行的训练方法为将每一天48个采样点中的前47个作为输入量，而最后1个作为其标签（label），在完成训练后在测试集上进行预测。此处以第1894#用户负荷为例，最终模型在测试集上预测负荷曲线及模型参数如下图所示：

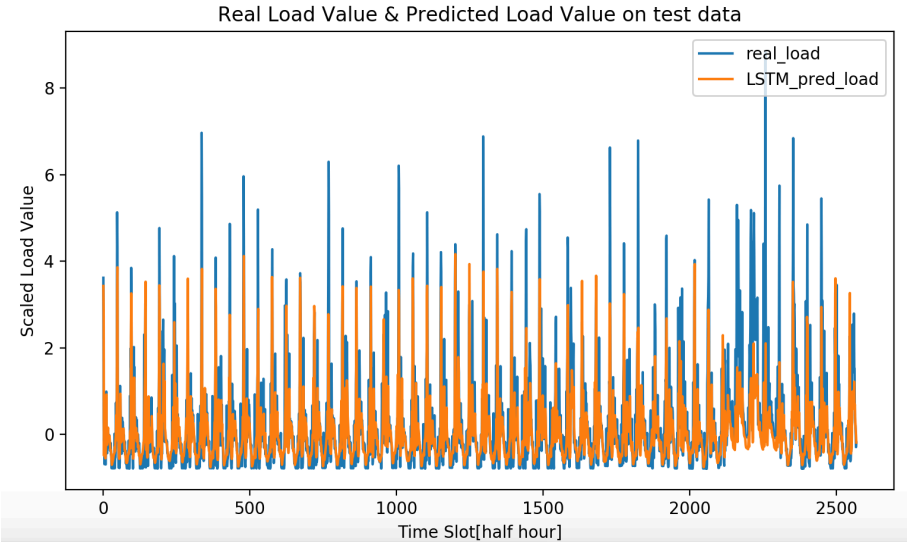

图 17 单表居民用户负荷数据集 LSTM 预测与真实负荷曲线对比图

由于单表用户负荷具有很强的日负荷周期性特征，每日用电量均会出现短期高峰而在大部分时间的归一化值为负数，因此LSTM负荷预测模型对此类波动剧烈的单用户负荷预测精度显然不如聚合负荷预测精度高。为了对比在这一负荷特征下预测模型的精度，选取了传统的BP神经网络与支持向量机预测模型在相通数据集上进行预测性能测试，这三类预测算法在测试集上的平均绝对误差mae和均方误差mse值分别如下表所示：

表 9 单表居民用户负荷数据集上不同用户预测模型性能对比结果

	LSTM 神经网络预测模型	BP 神经网络预测模型	SVM 支持向量机预测模型
平均绝对误差 mae	0.9869	1.1488	1.3772
均方误差 mse	0.6123	0.6801	0.8942

由实验结果表9可知，LSTM神经网络预测模型在测试集上的误差明显低于BP神经网络和SVM支持向量机模型，具备更加精确的预测性能，从而验证了在波动较为剧烈的负荷数据集上的预测精度性能。根据理论分析可知，传统BP神经网络由于不具有记忆机制，其网络泛化能力相对较差，训练时容易陷入局部最优；而SVM方法在处理大数据时间序列方面收敛速度较慢，并且预测精度相对较低。由于LSTM具有记忆功能并解决了长期依赖问题，因此即使对于负荷波动较为剧烈的单表负荷用户负荷数据，其短期负荷预测性能仍显著强于标准BP神经网络模型和SVM支持向量机模型。

3.5 本章小结

在本章节部分，详细介绍了将LSTM循环神经网络进行建模并在电力系统短期负荷预测问题的实现过程。首先阐述了本文中LSTM循环神经网络的实现的平台即Google TensorFlow机器学习框架的概况，之后即进行模型的数据集实验工作。首先通过数据预处理解决了历史负荷数据集中的缺失值异常值等“脏数据”问题，然后获得规范归一化历史负荷数据曲线。通过网络超参数调整优化得出LSTM循环神经网络模型的最优隐含层数量、每层神经元数量、优化器和学习速率等参数，然后将这一LSTM循环神经网络预测模型分别应用于区域聚合负荷数据集和单表居民用户负荷数据集进行预测精度性能测试和对比。在聚合式负荷数据集中采用的是在日前负荷数据基础上进行预测的方式，验证了对传统日前负荷预测值误差值的降低比例；而在单表居民用户负荷数据集采用的是历史真实负荷数据进行预测，并BP神经网络模型和SVM支持向量机模型进行了预测精度性能对比，根据性能对比验证了本文所选择的LSTM循环神经网络模型在解决短期电力负荷预测问题上的优越性，从而得出了本文所提出的预测算法和模型对于海量数据集上的的短期负荷预测具有较优的性能。

4 电力系统短期负荷预测模型的分布式并行计算实现

4.1 预测模型分布式集群架构概述

在前文工作中，我们已经通过构建深度学习LSTM模型实现了在集中式单机背景下的短期负荷预测工作，对聚合式负荷以及单表居民用户负荷两种不同特征的负荷数据集上的实验证明了这一模型比传统BP神经网络和SVM模型在预测精度性能上具有明显优势。然而随着当今电力系统大数据特征趋势的发展，数据量的攀升一方面意味着将有更多且内容更为丰富的历史负荷数据集能够用于数据挖掘和预测模型的训练工作，而另一方面也意味着TB甚至PB级别的历史负荷数据规模将无法通过单台计算机进行本地存储或完成模型训练和部署工作，因此在提升模型预测精度这一性能的基础上，本文将进一步提升预测模型在海量数据背景下的预测用时性能。

为了进一步提升模型性能，本文拟基于分布式并行计算架构提出LSTM电力系统短期负荷预测分布式并行计算模型。在现实应用中，海量负荷数据因其规模巨大的特点，往往无法通过单机进行存储或处理，需要借助分布式存储等手段加以存储，其中较为成功的应用即位Apache Hadoop分布式文件系统模型，可以有效解决超大数据集存储的问题。基于海量数据分布式存储后为了提升模型在海量数据下的训练和部署性能，本文拟基于Google TenosrFlow的分布式机器学习框架（Distributed TensorFlow）和Apache Spark分布式计算集群实现短期负荷预测模型从数据集读取、模型训练到模型部署全过程，以实现短期负荷预测分布式模型，从而从预测模型时间性能的角度大大提升模型在海量数据背景下的适用性。

首先本文将引出将前文所述的短期负荷预测模型变更为分布式集群部署的需要解决的主要问题。在前文工作中，我们已经明确了神经网络训练中主要的运算部分即为前向传播和误差反向传播的计算过程。在误差反向传播的过程中往往还需要调用一些优化算法如随机梯度下降或者前文训练中使用的Adams, Adagrad或RMSprop等，这些优化算法实际上就是在训练的每次迭代中先执行一次前向传播然后再执行一次误差反向传播。无论是传统BP神经网络，还是RNN循环神经网络或前文的LSTM循环神经网络，在其集中式模型中，前向传播时神经网络均会将一组训练样本作为输入，然后从输入层开始依次前向传播至每一隐含层，最终在输出层输出一个预测值。然后按照模型定义损失函数loss来计算预测误差，并将该预测误差从网络输出层依次反向传播到输入层，并且根据优化算法更新网络参数以下降误差。在足够多的训练轮数后使得损失函数降低到最小值并停止训练，确定最终的网络参数。根据这一人工神经网络训练本质，可以将前文所有人工神经网络在短期负荷预测中的运算抽象成如下的数学形式：

设输入的历史负荷数据集为D，损失函数loss为L，所训练的网络参数集记为θ，则神经网络在短期负荷预测模型训练过程可以认为是迭代执行更新方程，表达为如下迭代收敛算法：

$$\theta^{(t)} = \theta^{(t-1)} + \varepsilon \cdot \nabla_L\left(\theta^{(t-1)}, D^{(t)}\right) \tag{4.1}$$

上式中，t代表在网络参数集θ达到停止条件前的迭代次数，∇_L即为参数更新方程，通过计算损失函数L对于当前样本的$D_i(D_i \in D)$的梯度确定参数更新方向，而学习率ε决定了向参数更新方向移动的距离。如此一来由于导数可以随着数据样本进行累加，即$\theta^{(t)} = \theta^{(t-1)} + \varepsilon \cdot \sum_i \nabla_L\left(\theta^{(t-1)}, D_i\right)$。为了提升效率，所以一般每次迭代会同时输入一组训练样本即为$D^{(t)}(D^{(t)} \in D)$。

而在海量数据处理背景中，为了提升模型预测的性能使之适用于海量数据处理背景，本文采用数据并行手段予以解决。直观而言，该数据并行方法意味着将总的历史负荷数据集D切片并分发给一群计算设备组成的集群。对该集群的每一台机器进行编号为$p = 1,2,\ldots,P$。则分布式模型在第t次迭代时，由每个worker节点从其所分得的数据集切片中抽取一个batch的数据集，记为$D_p^{(t)}$并且独立计算梯度$\nabla_L\left(\theta^{(t-1)}, D_p^{(t)}\right)$。并且最后将所有worker节点所独立计算的梯度值相加并且用于总体神经网络的参数更新。因此对于高性能海量数据处理而言，其分布式模型训练过程应表达为：

$$\theta^{(t)} = \theta^{(t-1)} + \varepsilon \cdot \sum_{p=1}^{P} \nabla_L\left(\theta^{(t-1)}, D_p^{(t)}\right) \tag{4.2}$$

对于这一数据并行模型，每个worker节点均有其独立的数据切片，从而可以加快海量负荷数据集的训练速度，但同时该并行模型也需要每个worker节点在训练过程中一直读取和更新共享的网络权值参数集θ，因此需要实现worker节点之间的通信。通信也是分布式短期负荷预测模型的设计重点和难点，在本文中我们所选取和考虑的主要时参数服务器架构（parameter server architecture），即在所有worker节点之外独立设置一个参数服务器ps负责总的模型网络权值参数集的分发和更新工作。这一架构之间的节点通信情况可以由下图表示：

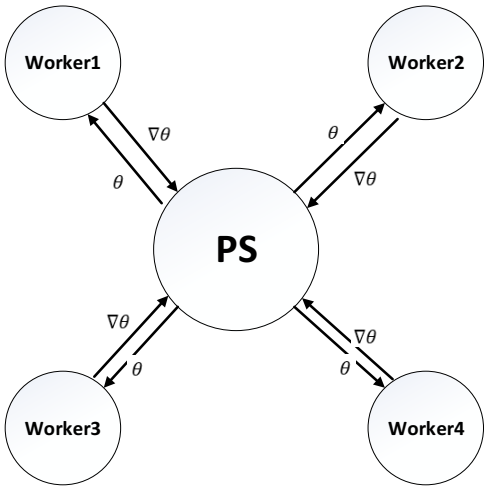

图 18 短期负荷预测模型分布式并行计算集群与通信关系图

对于该结构中的参数服务器ps（parameter server）而言，其在这一分布式架构中处于核心地位。参数服务器ps本质上是一个分布式的共享存储系统，可以提供系统层面上的数据并行模型的参数更新迭代抽象。一般而言，ps节点可以使每个worker节点独立通过网络

通信获取全局最新的模型参数集（即为上文中的参数集θ），在通信上该架构属于client-server（C/S）架构。根据这一分布式架构，本文的分布式短期负荷预测模型的训练过程实际上分为如下三步。首先，每个worker节点使用其数据切片集独立计算梯度∇_L，然后将梯度发送给参数服务器ps；之后参数服务器ps接收来自所有节点的参数并且进行累加，算出模型全局梯度值并进行全局模型共享参数的更新；最后需要设计一种合理的协调机制确保参数服务器ps节点和worker节点的严格时间一致性。

4.2 分布式 TensorFlow 机器学习集群架构

由4.1节可知，要实现模型的数据并行，需要构建如图18所示的集群结构。对于本文用于实现LSTM神经网络短期负荷预测模型的Google TensorFlow而言，实际上已经从2016年10月版本起便已经具备分布式能力，TensorFlow系统目前具有本地式(local)和分布式(distributed)两种实现方法，可以支持在海量数据处理背景中的模型分布式并行训练。

从TensorFlow架构角度而言，其本地式架构内包含有三种主要部分即client、master和worker在同一进程中。其中client具体负责定义TensorFlow主程序中的计算流图，并且启动session与master和worker进行通信交互，执行图计算。master具体负责根据程序在session.run()中指定的参数将计算流图分发给worker进行计算。worker具体调度其范围内的硬件设备(CPU或GPU，统称kernel)进行图运算[49]。

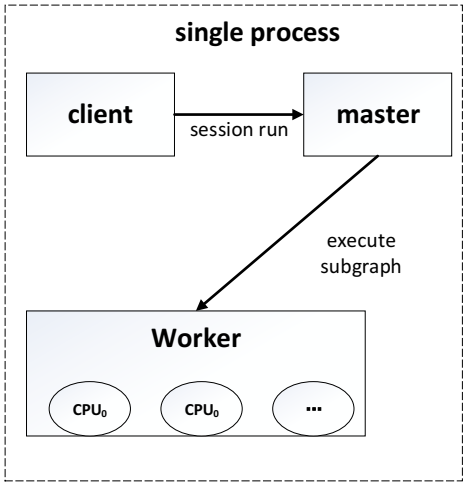

图 19 本地式 TensorFlow 运行架构

而对于分布式TensorFlow架构，则往往包含通过多机构成的集群结构，在这一结构中主要包含有如下术语：

表 10 TensorFlow 框架常用术语

术语名	含义
Client	是一个用于建立 TensorFlow 数据流图并创立与集群进行交互的会话 session 的程序，一个独立的 client 进程可以同时与多个 TensorFlow 的 server 相连，同时一个独立的 server 也可以与多个 client 相连。

Cluster	一个 TensorFlow 的集群 cluster 里包含了一个或多个作业(job)，每一个作业又可以拆分成一个或多个任务(task)，集群 cluster 对象可以通过 tf.train.ClusterSpec 来定义。
Job	一个作业(job)可以拆封成多个具有相同目的的任务(task)，比如说，一个称之为 ps 参数服务器的作业中的任务主要是保存和更新变量，而一个名为 worker(工作)的作业一般从事计算的任务。
Task	Task 对应一个特定的 TensorFlow server，属于 job 的一部分
Master service	是一个 RPC 服务，用于远程连接各设备，并且具体执行 session 会话，将数据流图分配给各 worker 执行。
Worker service	是一个 RPC 服务，以本地设备执行 TensorFlow 的数据流图
TensorFlow server	是一个运行了 tf.train.Server 实例的进程，其为集群中的一员，并有 master 节点和 worker 节点之分。

上述各主要构成部分在分布式TensorFlow架构中的位置如下图20所示：

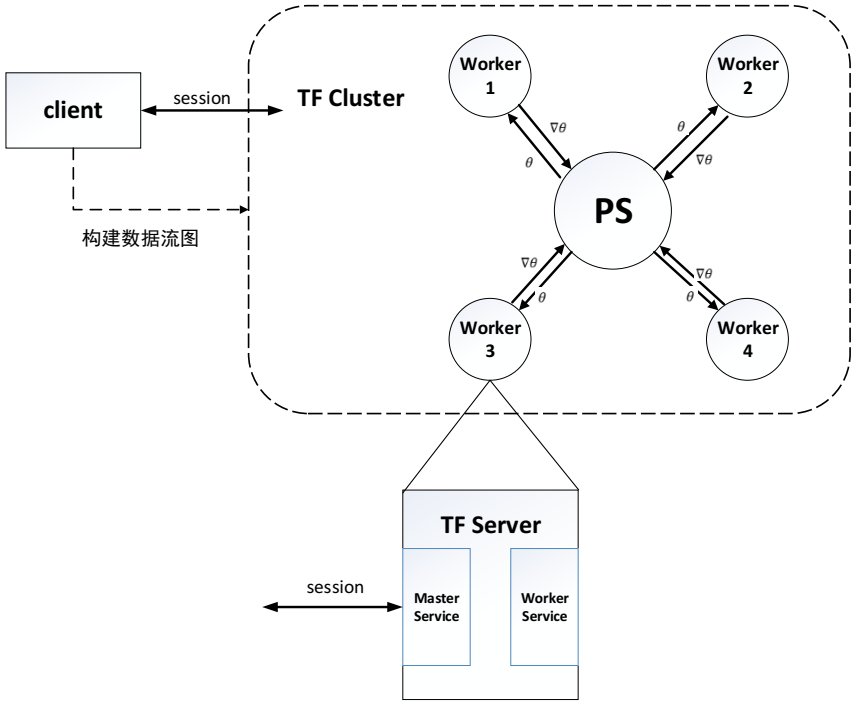

图 20 分布式 TensorFlow 运行主要架构

目前，分布式TensorFlow主要支持两类分布式拷贝方式，分别称为图内拷贝和图间拷贝。对于图内拷贝而言，将在client客户端建立一个独立的计算流图，通过ps作业(/job:ps)来声明计算流图中的各节点node，并且由worker作业(/job:worker)来具体执行，即计算流图的具体计算任务。在图内拷贝分布式情况下，实际上每个worker作业并不能获取完全的计算流图，而只能执行例如数据预处理、Loss损失函数计算和误差反向传播等计算任务，只有client客户端程序才存有完整的计算流图。在具体执行数据流图计算的过程中，实际上会由master首先将输入数据集根据worker数量加以均分，worker进程执行自己的计算子图（subgraph）得到部分计算结果，最终由在在client客户端所在计算机上对所有worker执行的部分计算结果加以汇总，计算出总的训练结果值（例如损失函数Loss），再由整体的优化器进行网络参数优化。在子图划分的过程中，对于总计算流图中每一因子图划分而截断的边，由master

负责向截断产生的两个子图对应位置上分别插入一个发送节点（send）和一个接收节点（recv），使得分布执行的task之间能够通过发送-接收通信机制进行相互的数据传输，不至引发子图计算的中断。由于此时完成的模型计算流图实际上分布在多机上，因此图内拷贝又称为模型并行模式。显然，图内拷贝模式更适用于神经网络模型巨大、参数众多以至于单机无法存储的情况。

除了图内拷贝以外，分布式TensorFlow的另一种模式称为图间拷贝，即在每个worker task上都会实际运行有一个独立的client客户端，并通过客户端单独声明每个任务(/job:worker)。在这一分布式形式下，实际上每台机器都会使用完全相同的计算图，但分别负责计算不同的输入数据batch。该分布式形式中包含的参数均通过ps (/job:ps)进行声明，因此实际上图间拷贝属于数据并行。显然，图间拷贝模式节省了在训练过程中数据流图本身的通信时延，实际上更有利于进行超大规模集群的部署和利用。

此外分布式TensorFlow还支持有同步训练（Synchronous training）和异步训练（Asynchronous training）两种分布式训练方式。在同步训练模式下，所有的计算task都会读取当前ps作业中的相同参数值（如LSTM网络的网络权值集）用于并行化梯度计算，并且要等待所有机器都完成了梯度计算工作后，最后合并梯度计算结果并统一更新模型参数值；而异步训练模式下实际上将由每台机器独立进行训练过程中的梯度计算，且一旦计算完成后立刻更新到ps作业中，而不必等待其他机器。根据官方文档实验，采用同步并行训练时模型可达到的精度往往更高，但其计算过程中由于需要等待机器，因此速度取决于计算速度最慢的一台机器，可能出现木桶效应拖慢模型训练的速度。

由于本文中所构建的LSTM短期负荷预测模型的网络参数量规模适中，完全可以在单机容纳整个TensorFlow数据流图，因此从简化模型通信负担的角度，本文所进行的模型分布式并行计算实现均采用图内拷贝方式，即实现数据并行，在集群内每个节点上均执行相同计算流图而执行不同的数据batch进行训练，验证分布式并行计算及不同的并行训练模式对模型训练的性能影响。

4.3 Apache Spark 分布式计算集群架构

Spark是Apache软件基金会组织所发布的开源集群计算系统，可以在集群上实现分布式内存计算，具有多用途、高速率、快速处理等特点，提供了Java, Scala, Python和R等多种语言的高级API，具有支持通用执行图计算的计算引擎[50]。基于Spark编写的Spark应用在集群上作为独立的进程组来运行，通过在main主程序的SparkContext完成协调，称之为driver驱动程序。为了在集群上运行，SparkContext可以连接到几种不同的Cluster Manager集群管理器，包括Standalone, Mesos和YARN等，并通过这些集群管理器来分配集群中的内存和计算资源。进行集群网络连接后，Spark即能够获取集群中节点上的Executor，从而利用该进程

运行计算或存储应用数据。最后由Spark将应用代码发送给Executor，并由SparkContext发送Task到Executor来执行应用代码。上述Spark运行结构及执行流程可以由下图21进行表示：

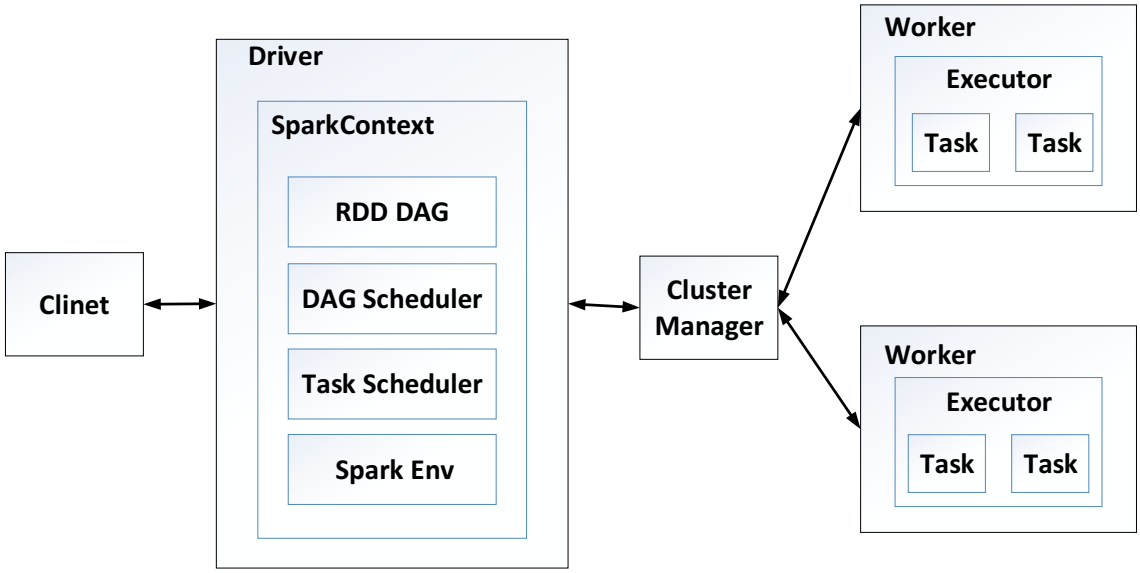

图 21 Apache Spark 分布式计算框架结构图

Spark分布式计算集群的基本构成包含以下术语，可与图21和图22以对应：

表 11 Apache Spark 框架常用术语

术语名	含义
Application	是用户在 Spark 上构建的程序，由集群的一个 Driver 程序和多个 Executor 组成
Driver	用于运行 Spark 程序上的 main()方法并且创建 Spark Context
Executor	是在 worker 节点上启动并运行 task 的进程，可以将数据保存在内存或硬盘
Cluster Manager	是在外部的用语获取集群上资源服务的管理器，例如 standalone,mesos 或 YARN
Deploy mode	分为 Cluster 模式和 Client 模式，在 Cluster 模式下框架会在集群内部启动 driver 程序，而 Client 模式下则由 submitter 在集群外部启动 driver 程序
Worker	集群中用于运行应用代码的节点
Task	具体发送到 Executor 中的工作单元
Job	是由多个 Task 所组成的并行计算程序，可以从 spark action 中获得响应
Stage	是 task 组，由 Job 拆分而成，且 Stage 之间互相依赖

Spark 中的分布式执行是通过将这种有向无环图DAG编译生成的stage分割到不同的机器上执行的，从而构成了Spark的集群结构。驱动器（Driver）除了包含前文单机模式基本的Spark Context以外，在分布式下还要有两个调度器（Scheduler）组件——DAG 调度器和任务调度器，并且还要将任务对应到worker节点。

在Apache Spark中，对数据的运算实际上抽象为有向无环图DAG(directed acyclic graph)，该有向图中每一个顶点都代表RDD(Resilient Distributed Datasets) 弹性分布式数据集，而每一条有向边都是对RDD的运算。RDD是分布式内存的抽象概念，对真实的来自HDFS等来源的文件进行了封装，在Spark操作文件过程中可以将RDD视为Spark的一个对象，其本身运行于内存中，无论是文件读、操作还是计算，所得结果都仍是RDD。Spark的一个RDD运行任

务可分为两部分Transformation和Action来执行，其中Transformation主要包括map, flatMap, filter, union, sample, join, groupByKey等操作，可以用来定义一个新的RDD，而Action主要为具体运算，包括collect, reduce, count, save等，操作返回一个结果。在进行Spark运算的过程中，需要将本文中的短期负荷预测模型训练过程建模成前文所述的有向无环图，从而在RDD上进行Transformation和Action。在具体运行过程中，该有向无环图需要被编译成stage，作为一系列并行运算的task组来执行。下图22为Spark对的一个分布式运算示例，其中深蓝色部分为RDD分区，连接即为Spark对RDD进行的Transformation操作。

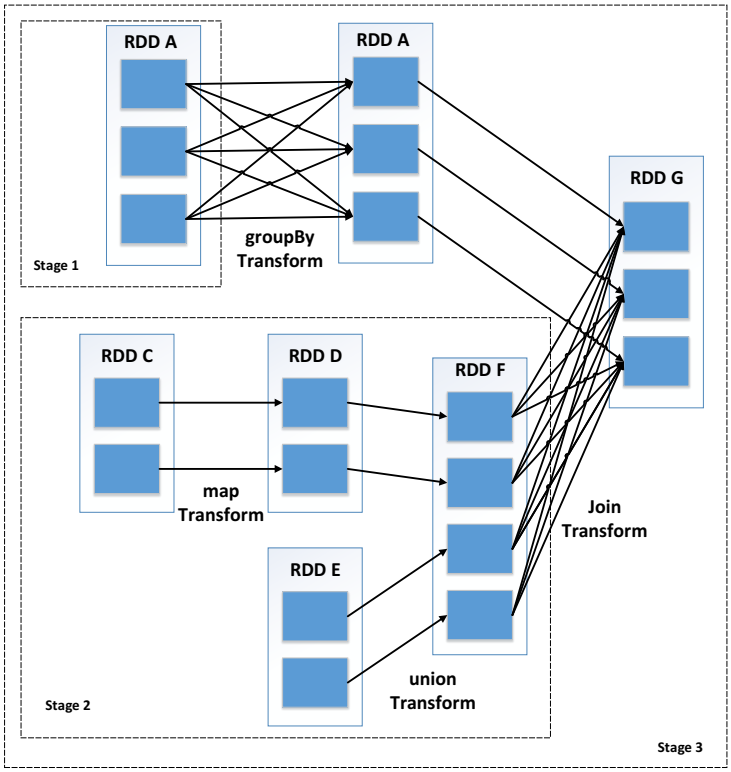

图 22 Spark RDD 运算逻辑示例

4.4 TensorFlow 与 Apache Spark 集群的融合

根据4.2节可知，通过合理搭建TensorFlow的分布式集群实际上就可以实现LSTM神经网络模型的分布式并行训练以加速预测模型的训练速度。然而对于实际负荷预测工作而言，历史负荷海量数据往往体积庞大，超出单机存储能力，因而需要通过Hadoop分布式文件系统（HDFS）进行分布式存储。HDFS具有较高的容错性，十分有利于大规模数据在多个存储设备上进行存储。而对于短期负荷预测模型训练，其对网络参数的更新和保存需要大量迭代计算，因而HDFS涉及的大量硬盘读写操作会制约模型训练速度，而Spark中所采用的RDD弹性分布式数据集较好的解决了这一问题，通过分布式内存抽象大大加快了读写速度，因此目前的海量数据存储和基本处理往往采用Spark+Hadoop集群进行。

由于海量负荷数据的实际存储采用Spark+Hadoop集群，而深度学习神经网络模型的分布式训练采用TensorFlow集群，两类集群往往在物理上属于不同的计算机实现，导致实际进

行短期负荷预测工作时可能需要首先在Spark+Hadoop集群上准备好数据集，之后通过网络通信将数据集传送到TensorFlow集群上，在集群完成训练后再将满意的LSTM短期负荷预测模型重新传送回Spark+Hadoop集群，并最终在Spark+Hadoop集群部署和实现预测工作。通过上述业务流程我们可以很清晰的意识到该系统目前运行较为复杂，并且数据集通过网络的传输存在较多的端对端时延，其问题如下图23所示：

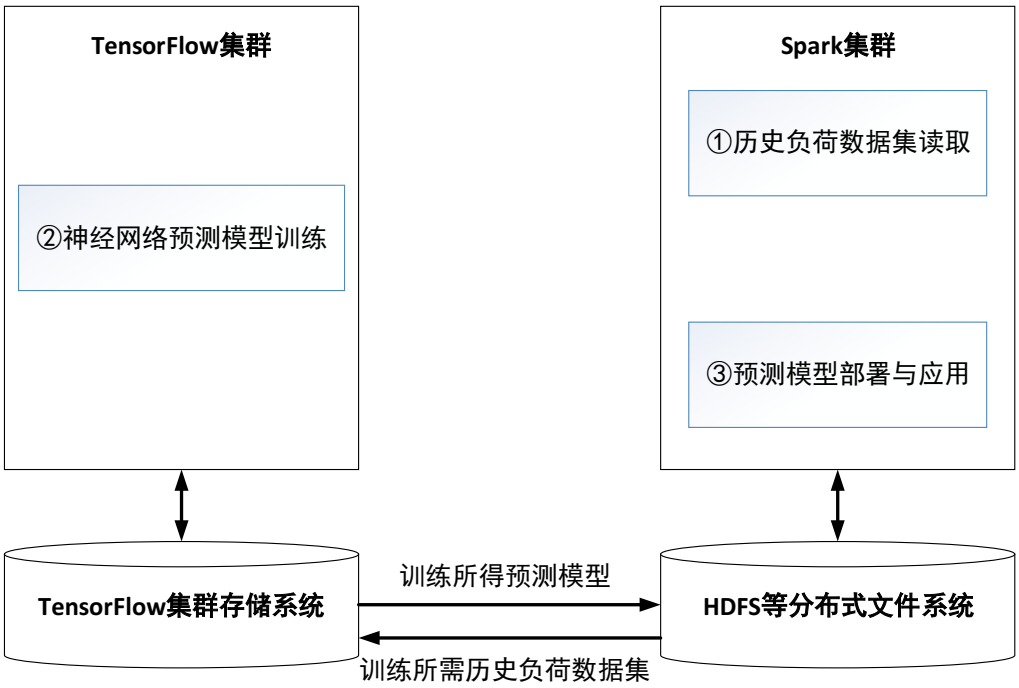

图 23 融合前数据集存储集群与预测模型训练集群分离部署情况

这一部署方式在未来海量数据处理背景下，难以对短期负荷预测数据进行及时的处理和部署，因此在本文最后一部分将考虑采用Spark分布式计算集群与TensorFlow分布式机器学习集群相融合，即让上图中分离的两块集群融合为整个集群，让短期负荷预测分布式模型可以真正实现在数据存储集群上进行模型训练与模型部署，以保证预测模型的时间性能，满足未来短期负荷预测高性能海量数据处理的要求。

本文所设计的融合架构主体仍为Spark集群，而令TensorFlow在这一集群上分布式运行，这一架构称为TensorFlowOnSpark，可以为每个Spark集群计算节点提供TensorFlow运算能力，具体思路如下图所示：

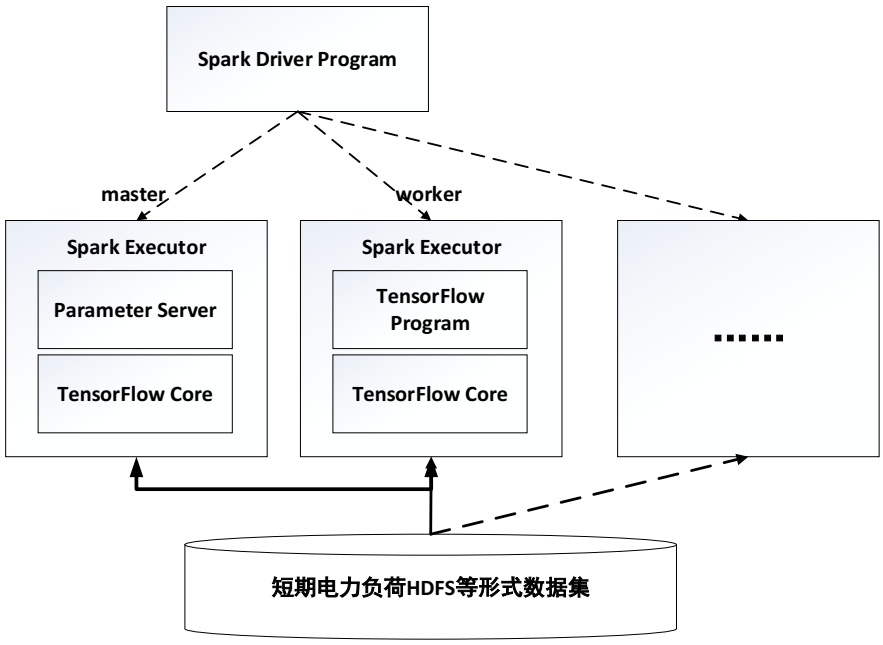

图 24 融合后在 Apache Spark 集群上部署 TensorFlow 分布式集群架构图

在架构上，将TensorFlow分布式集群中的worker与Spark集群中的Executor进行一一映射，即在Spark集群中的Executor节点上执行TensorFlow的具体任务，让TensorFlow分布式运算在Spark集群中运行。本文融合架构将TensorFlow的算法和TensorFlow核心部署于同一个Spark Executor中，并让TensorFlow任务能够通过TensorFlow的feed_dict向计算流图节点投喂Spark RDD数据，执行TensorFlow程序进行神经网络参数训练。

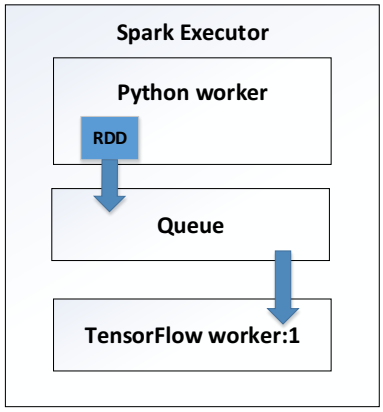

图 25 SparkRDD 数据集在 worker 节点上的输入过程

对于每个Spark Executor而言，若其在计算集群中为ps参数服务器节点，则其只运行python worker进程并执行ps工作，负责原地停等等待通信连接更新参数即可；而对于worker计算节点而言，其上将通过python worker执行RDD.mapPartitions()方法将每块RDD分区先输入到一个queue队列内（见图25），然后通过队列向TensorFlow通过TensorFlow主程序中的feed_dict()方法输入具体训练数据，并将模型输出结果也返回到该队列内。

综上所述，在完成Spark集群与TensorFlow集群的融合后，对于本文所提出的LSTM电力系统短期负荷预测模型可以真正在同一集群上分布式并行实现。这一分布式并行计算模型

可以划分为负荷数据集处理和预测模型训练两部分，在实际部署时首先启动Spark集群，然后将存储于集群上的HDFS等形式的历史负荷数据集通过Spark的sc.textfile()函数转换为RDD数据集形式，之后向集群中的各节点分发TensorFlow主程序及相关程序依赖，并在计算流图节点以feed_dict投喂RDD历史负荷数据集在集群各节点执行分布式模型训练，在训练完毕后可以直接在集群实现预测模型的实际应用和集群关机等功能，如图26所示：

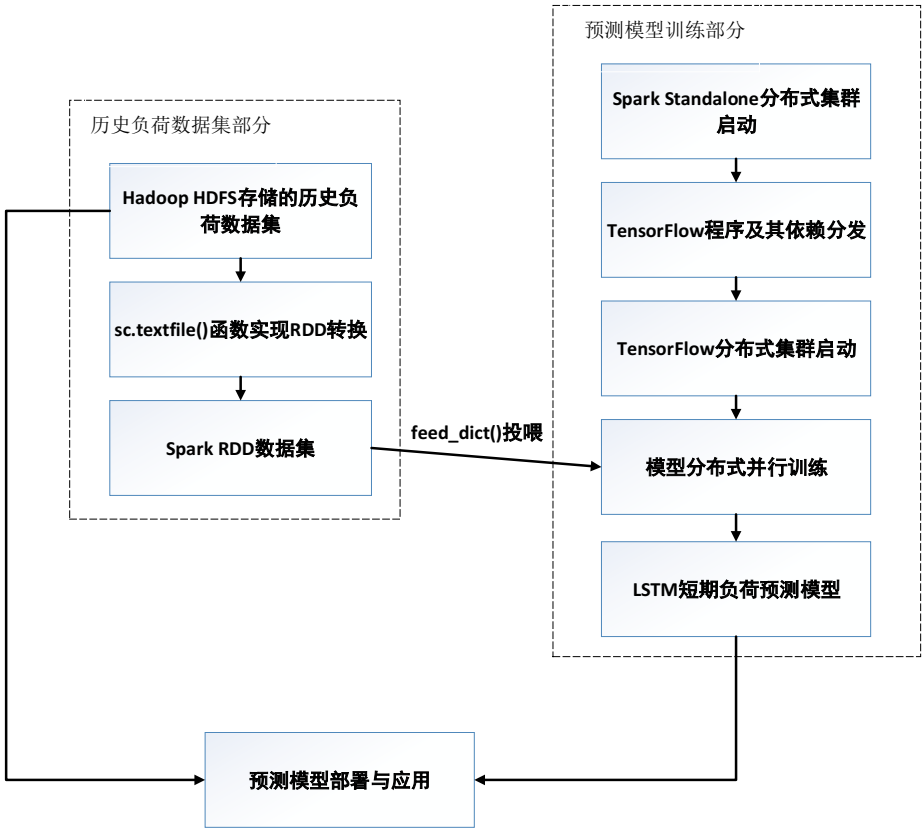

图 26 本文短期负荷预测模型的分布式并行执行流程

4.5 LSTM短期负荷预测模型的分布式并行计算实现

4.5.1 实验软硬件环境

根据前文所述基于Spark集群的TensorFlow分布式结构进行实验环境搭建，实验验证在这一集群上部署前文所提出的LSTM短期负荷预测模型的有效性。则对于大规模历史负荷数据集，本文利用Apache Hadoop的HDFS分布式存储系统模拟实际现场的分布式历史负荷数据集存储，并且在同一集群上实现上文所述的TensorFlowOnSpark融合架构，因而实现这一LSTM短期负荷预测模型的分布式并行计算架构总体情况如下：

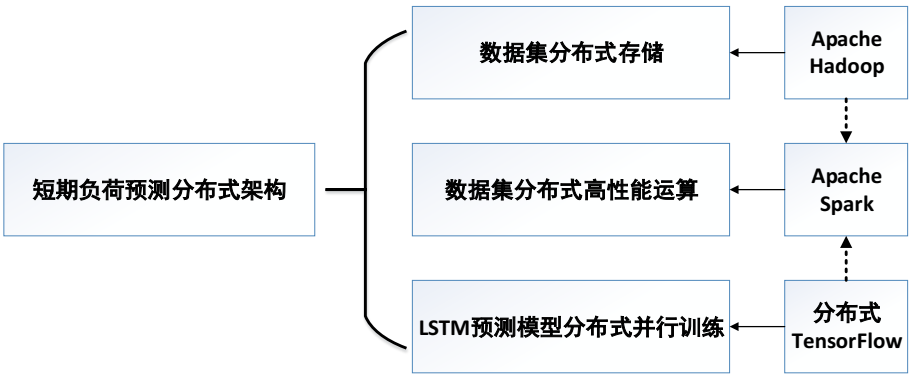

图 27 预测模型的分布式并行架构

实验环境在Macbook Pro电脑通过VMWare Fusion搭建虚拟机集群，集群设备的软硬件配置如下表12所示：

表 12 实验平台软硬件环境配置情况

集群设备	配置
CPU	Intel(R) Xeon(R) CPU i7 4850HQ/4870HQ
CPU 频率	2.2GHz
内存	1G
硬盘	1G SSD
操作系统	Linux Ubuntu 18.04
Scala	2.11.8
Java JDK	1.8.0_171
Hadoop	2.7.6
Spark	2.2.6
TensorFlow	1.8
Python	2.7

4. 5. 2 分布式集群的搭建

为了实现TensorFlow并行分布式与Spark分布式框架结合的LSTM神经网络训练，本文实验首先搭建分布式集群环境。首先需要在分布式集群上搭建Hadoop集群和Spark集群，从而同时使用Hadoop集群的HDFS文件系统实现海量负荷数据的存储，缩短磁盘数据到内存的读入时间，并利用Spark RDD的优势运行TensorFlow程序。

为了模拟实际应用环境，本文利用笔记本电脑安装VMware Fusion创建了4台虚拟机用来集群搭建。在4台虚拟机中设置1台为主节点（master），另外3台作为从节点（slave）。在每个节点上均安装和部署Hadoop, Spark进行安装，并在集群节点间进行SSH配置以确保相互间都可以做到免密访问。此外集群下每台机器的配置情况如下表13所示：

表 13 实验分布式集群搭建配置情况

IP	Hostname	Hadoop 节点	Spark 节点	部署情况
172.16.3.132	master	namenode	Master	Hadoop,Spark,TensorFlow
172.16.3.135	slave1	datanode	Worker	Hadoop,Spark
172.16.3.136	slave2	datanode	Worker	Hadoop,Spark
172.16.3.137	slave3	datanode	Worker	Hadoop,Spark

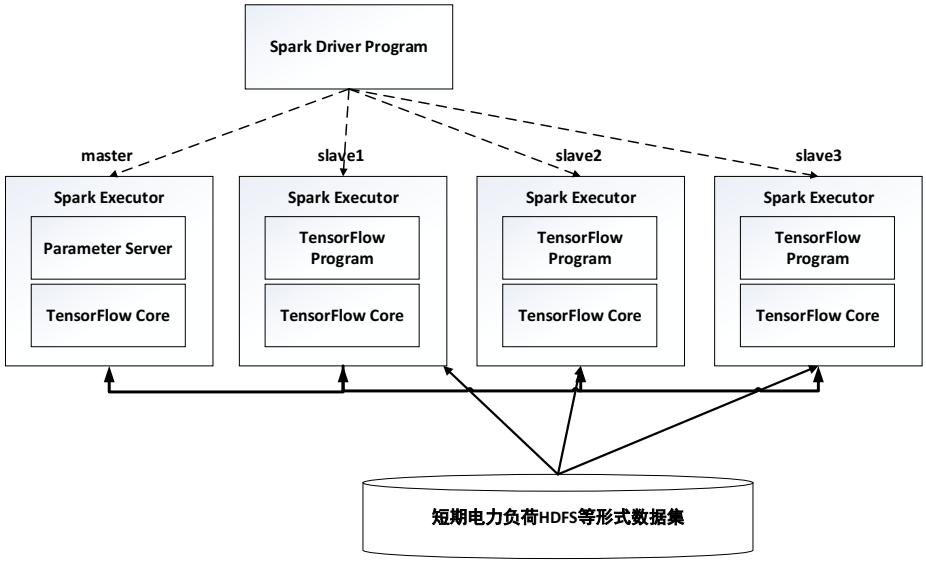

图 28 实验分布式集群配置情况

在以上的虚拟机集群配置中，Hadoop集群中的namenode节点虚拟机master将在运行期间管理集群HDFS文件系统的文件目录树，以及目录树中所有文件和文件夹的短期负荷数据集索引，而不会直接进行历史负荷数据集的存储。datanode节点虚拟机slave1和slave2则为集群中HDFS文件系统中真正执行数据实际存储的节点，负责存取分块的历史电力负荷数据集，并定期向namenode汇报数据分块的列表。

Spark集群中的Master节点虚拟机master负责整个Spark集群中设备的控制，在Master节点上将执行Driver驱动程序进而负责具体任务的分配。而集群中其他的Worker节点负责装载数据，执行具体处理计算，并返回结果，只负责计算工作，并受到Master节点的管理。在本文配置中，我们将虚拟机master同时设为Master和Worker节点，则其可以同时进行集群的管理和计算。

根据前文所实现的TensorFlow与Spark集群的融合，在本虚拟机集群中将由master上负责运行TensorFlow分布式集群中的ps job，即作为参数服务器ps在预测模型分布式并行训练过程中提供模型参数动态存储和通信交互。而所有的从节点slave上将由master分发TensorFlow程序及其依赖包（包括Python2.7、TensorFlow核心及架构融合文件）进行具体模型训练的计算任务。

在完成上述虚拟机集群搭建后进入Hadoop安装目录，在内部的/sbin路径下有“start-a11.sh”和“stop-a11.sh”两个脚本程序，分别用于启动和停止所有集群节点上Hadoop运行的java进程，即启动HDFS分布式文件系统。本文中在启动Hadoop集群后可以通过50070端口在浏览器访问Hostname master上的Hadoop web UI管理界面。Hadoop web UI提供最基本的集群监控功能，通过该监控界面，能够分别查看该Hadoop系统中的namenode运行情况以及datanode的位置、状态信息以及存储和内存使用情况，并且可以查看目前Hadoop系统中所拥有的HDFS文件情况。将本文实验所使用的负荷数据集上传到HDFS，模拟实际工作中在各调度中心或主站的主机上存储的大规模历史负荷数据集。

之后启动Spark集群，其集群管理器本文中使用Standalone模式，可以通过在Spark安装目录内部的/sbin路径下有“stan-a11.sh”和“stop.a11.sh”两个脚本程序，分别用于启动和停止所有集群节点上Spark运行的java进程。类似地，在启动Spark集群后可以通过7077端口在浏览器访问Hostname master上的Spark master web UI监控界面查看集群中活动的 worker 数量，集群CPU核心数量和占用情况等。在启动Spark集群后，在Python环境中可由pyspark启动SparkContext，并调用sc.textfile()方法从Hadoop HDFS文件系统中读取用于模型训练的历史负荷数据集转为Spark RDD内存存储形式。上述Hadoop HDFS与Spark RDD准备代码如下：

表 14 历史负荷数据集 Hadoop HDFS 上传与 Spark RDD 转换代码段

```
$hdfs dfs -mkdir /user/${USER}/
$hdfs dfs -put ${HOME}/load_data.csv /user/${USER}/lstm/load_data.csv
$pyspark
from pyspark import SparkConf, SparkContext
conf = SparkConf().setAppName(“textfile_dir”)
sc = SparkContext(conf=conf)
datafile = sc.textFile(“hdfs:///user/${USER}/lstm/load_data.zip”)
sc.stop()
```

由于Spark启动集群运行的程序需要预先编译打包，然后将TensorFlow程序和依赖包分发至集群，因此本文需要制作python、TensorFlow和TensorFlowOnSpark三部分的依赖包并上传至集群，确保打包上传的分布式短期负荷预测程序能够在所有集群上执行。此外将前文LSTM短期负荷预测的TensorFlow程序进行分布式变更，嵌入TensorFlowOnSpark融合集群的相关代码段，生成python程序文件为lstm_dist.py。此外还需要定义在这一集群上运算所需要的各参数指定，相关制定情况通过argparse方法向程序传入相关控制参数，最后在Spark界面中通过Spark-submit提交任务，指定集群计算节点数量和内存容量等参数并开始执行预测模型训练任务，其提交代码段如下表15所示：

表15 Spark集群上API提交运行TensorFlow分布式并行模型训练任务代码段

```
${SPARK_HOME}/bin/spark-submit \
--master ${MASTER} \
--deploy-mode cluster \
--queue ${QUEUE} \
--num-executors 3 \
--executor-memory 2G \
--py-files ${HOME}/TensorFlowOnSpark/tfspark.zip,${HOME}/TensorFlowOnSpark/lstm_dist.py \
--conf spark.dynamicAllocation.enabled=false \
--archives hdfs:///user/${USER}/Python.zip#Python \
```

```
--conf spark.executorEnv.LD_LIBRARY_PATH="$JAVA_HOME /server" \
${HOME}/TensorFlowOnSpark/lstm_spark.py \
--mode train \
--model forecast_model
```

在完成预测模型训练后，在测试集上计算预测性能，需要变更模型代码中的mode之后再次向集群提交预测计算任务，代码段与上表15类似，故不赘述。

4.5.3 不同集群规模与并行模式下的预测模型性能对比

为了实现对实际生产中的海量数据处理支持，本文选择了采用对网络通信依赖相对较少的TensorFlow图间拷贝（Between-graph replication）模式在上述融合集群上进行预测模型实验，对比不同集群规模与并行模式的模型性能。为了对训练方式加以对比，本文同时采用了同步训练和异步训练两种方式进行模型并行实现，并且通过调整集群中的节点数量进行训练比对。训练数据集仍为前文试验中的欧洲互联电网（ENTSO-E）瑞士聚合负荷数据集，但由于硬件条件的限制，在原模型的基础上需要将batch_size调整为1，即采用在线学习，以最大限度提升模型精度，区分各分布式部署模式下的模型训练性能，模型训练轮数epoch均设置为100，此时模型训练集中共有14592条数据，验证集中共有3648条数据。在根据上文所述步骤修正原程序代码部署为分布式后，启动Hadoop分布式文件系统以及Spark计算集群，通过表15所示spark-submit脚本提交计算任务进行预测模型训练，记录各并行模式下模型训练用时情况如下表16所示：

表 16 预测模型不同并行模式下模型训练时间表

分布式部署模式	训练耗时/s
单机模式	1326
双机异步训练	712
双机同步训练	728
三机异步训练	491
三机同步训练	527

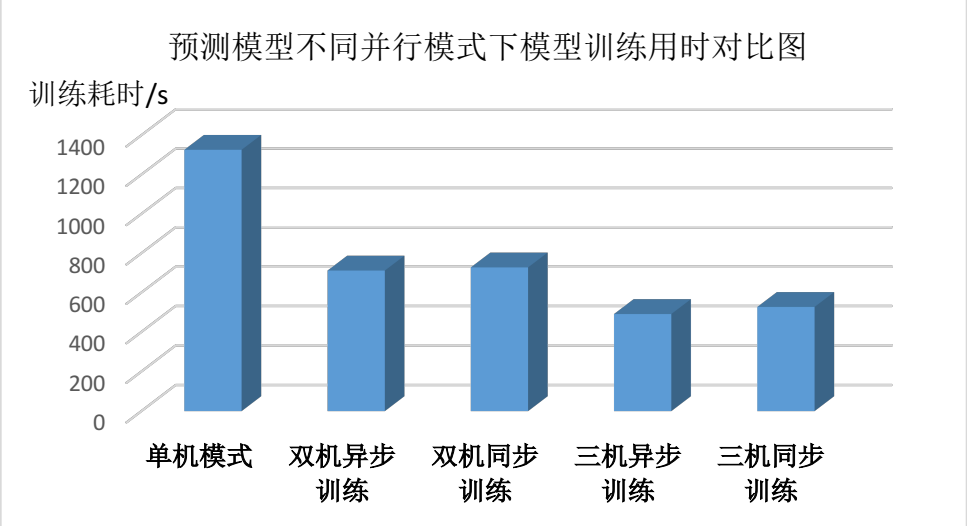

图 29 分布式部署模式与模型训练时间关系图

由以上计算时间对比图29可知，随着集群中运算设备的增加，模型训练时间均大大减少，其中双机异步和同步训练时间相较于单机均减少接近50%，而三机异步和同步训练时间均相较于前文模型减少约60%，训练时间减少比例大致接近于线性，可以证明通过并行计算确实可以显著提升短期负荷预测模型在海量数据处理背景中的应用情况。

另一方面，从本文实验角度而言，可以发现无论是双机还是三机训练的情况下，模型同步和异步训练区别并不显著，三机异步模型训练时间略短于三机同步模型。根据初步分析，认为这主要是由于两方面的因素导致。其一是本实验中所有模型均运行在配置完全相同的虚拟机集群上，这意味着所有的机器在参与这一相同数据集的计算时，其性能和运算速度大致相同，使得同步并行并不存在显著的“短板”效应，集群硬件配合较好确保了总体时间的接近线形减少。此外，对于同步并行而言，其所有worker节点需要在每一次迭代向参数服务器ps节点发送更新的权值并且等待参数服务器对权值更新完毕重新传输，因此会产生更大的网络通信代价，这一代价在本实验的三集群模型中并不明显，但训练时间上的细微差异仍可以体现出同步并行的这一不足，受制于通信手段和网络带宽的限制，这一模型在实地海量数据背景下进行工程应用时，必须对同步并行的这两方面制约因素予以考虑。

除了预测用时性能外，本文同样对不同分布式部署模式下模型在测试集上预测精度性能进行了比较，以探究在执行相同训练轮数的情况下不同分布式部署模型对所得预测模型的预测精度性能的影响。所测试得到的测试集上均方误差mse分别如下表17所示：

表 17 不同分布式部署模式下模型在测试集上预测精度性能表

分布式部署模式	模型预测均方误差 mse
单机模式	0.1921
双机异步训练	0.2011
双机同步训练	0.1937
三机异步训练	0.1989
三机同步训练	0.1968

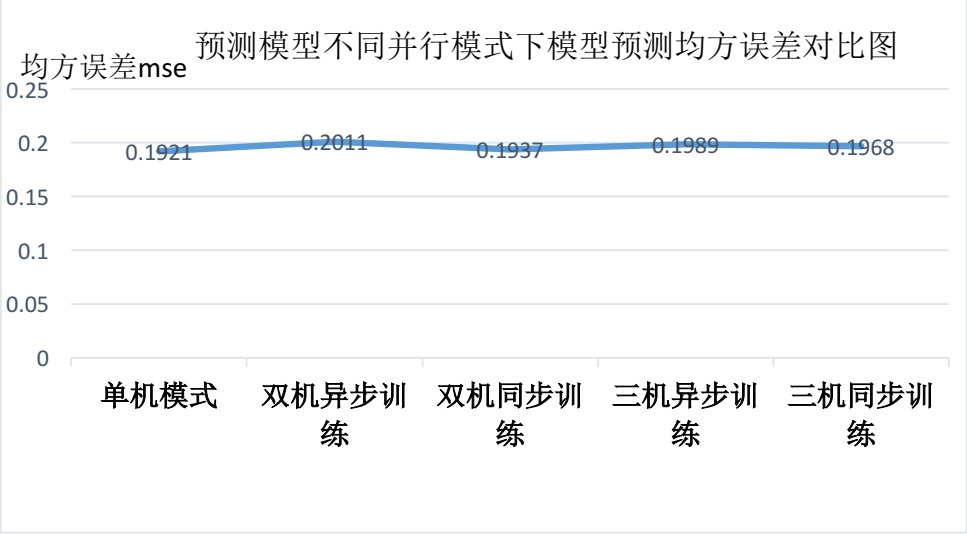

图 30 预测模型不同并行模式下模型在测试集预测均方误差对比图

从模型预测精度的角度，按照模型在测试集上的均方误差（mse）而言，各模型训练所达到的最后误差水平在模型训练随机误差范围内基本一致，基本都与单机模式训练所得模型的测试集均方误差一致。同步训练的模型精度比异步训练模型略有提升，反映出在数据集和模型训练参数不大的情况下，实际上模型并行化以及并行方式对于最终的训练目标区别不大，训练时间是模型并行化的主要制约因素。

4.6 本章小结

在本章主要阐述了本文对前文所搭建的电力系统短期负荷预测模型在海量数据处理背景中应用的性能优化模型，即通过实现模型的分布式并行训练提升算法的海量处理性能。本章首先介绍了分布式并行计算模型的一些基础结构，然后介绍了本文实现这一分布式模型训练所基于的Apache Spark分布式计算集群以及Google TensorFlow上目前已经支持的分布式机器学习部分，然后将这两部分的框架进行融合实现TensorFlowOnSpark框架，从而实现在同一集群上从数据存储、模型训练到模型部署的全过程分布式部署。之后介绍了本文实现这一分布式预测模型的实验，通过搭建虚拟机集群并在原单机程序中添加指定分布式参数进行实现。利用所搭建的分布式LSTM循环神经网络电力系统短期负荷预测模型进行了性能测试，考察包括集群中不同计算节点数量以及并行训练模式对预测模型用时和精度性能的影响，从而验证了该分布式并行计算架构对于海量数据背景下的电力系统短期负荷预测的优越性能。

5 总结与展望

5.1 本文工作总结

近年来，随着传统的电力数据向智能电网电力大数据的演变，各种海量输出处理算法在电力系统的应用愈发广泛。根据文献[51]中的测算，美国PGE公司每个月从900万个智能电表中收集超过3TB的数据，而我国国家电网公司的2.4亿块智能电表每年产生的数据规模超过200TB，因此通过研究高性能算法应对海量数据背景下的挑战，对确保电力市场的健康运行和整体能源系统的效率提升具有不可估量的深远意义。作为智能电网实时调度的基础，短期负荷预测算法从海量数据中提升精度以及对海量数据的快速处理性能更是电力大数据中的关键点，也是本毕业论文全文的着眼点。为了实现在海量数据处理背景下的精准高效短期负荷预测，本文主要做了如下两个方面的工作：

首先，本文应用深度学习领域中的长短期记忆循环神经网络LSTM模型到电力系统短期负荷预测模型中，利用这一神经网络结构对大规模时间序列数据处理和训练的有效性实现了其在短期负荷预测领域的应用。LSTM循环神经网络作为深度神经网络中最适用于序列数据预测的一种，已经成功应用于自然语言处理及机器翻译等应用场景。由于电力负荷数据本质上可以类比为随机非线性序列，因此文本利用这一模型解决短期负荷预测问题完全合理。本文从收集的实际负荷数据集出发，利用LSTM循环神经网络模型对聚合负荷和单表用户负荷两种不同数据特征的序列数据进行预测和性能分析，并与传统的负荷预测模型BP神经网络、支持向量机等进行比对，结果证明LSTM循环神经网络模型在聚合负荷和单表用户负荷两个规模层面上均具有精度上的优越性，并且可以解决海量数据的处理和预测问题，其应对平稳连续性相对较好的聚合负荷数据预测性能优于随机非线性较强的单用户负荷数据。

其次，本文将TensorFlow分布式机器学习集群与Apache Spark分布式计算集群相融合，提出短期负荷预测模式的分布式并行计算架构，在分布式集群上实现了LSTM循环神经网络模型的分布式并行训练和模型预测，该分布式模型可以通过Spark提交计算任务及模型而在Spark计算集群上启动TensorFlow集群实现分布式计算，从而避免了在海量数据背景下数据库存储、预测模型训练和实际预测模型部署三部分分离导致通信时延的问题，实验中利用TensorFlow本身的图间拷贝模式实现了数据并行的并行化模型训练，可以任选以同步或异步方式实现模型参数训练。通过这一融合架构，可以实现电力大数据背景下历史负荷数据集的分布式存储、预测模型的并行训练加速以及预测模型的快速部署与应用，从而给出了海量数据背景下短期负荷预测问题的一个具有参考价值的优化方案。

5.2 模型适用范围分析与未来优化方向

本文中所提出的模型和算法对于解决海量负荷背景下的短期电力负荷预测具有一定的学术价值和实践意义，但也应清晰认识到受限于具体实验硬件设备和实验数据集，本文所提出的方案仍具有一定的适用范围和局限性，该课题还有如下几个部分的工作需要进一步探究和拓展，我将在未来深造过程中予以进一步研究。

从适用范围的角度，从3.4节的实验论证可知，本文所提出的LSTM深度学习模型在短期负荷预测领域对于平稳性较好的负荷数据（如区域聚合负荷数据）预测精度表现优于非平稳随机性强的负荷数据（如单表居民用户负荷数据），这也是该模型对时间序列长程依赖的特性决定的，因此本文所提出的LSTM预测模型更适用于负荷数据平稳且连续的情况。

此外对第4章的分布式并行计算架构，其卓越的海量数据处理加速性能只有在数十G以上规模的数据集分布式存储情境下才能得到充分凸显，而实现分布式并行计算会导致计算模型较为复杂，程序中需要指定各种操作的具体节点等，因此可知在数据集规模较小（如县级调度中心）的情况下仍应优先考虑采用单机模式进行计算和预测。

从未来优化方向的角度，本文的局限性和下一步的研究工作着眼于两方面：

（1） 对于预测模型训练数据集的优化

在本文实验所使用的数据集均为开源的历史负荷数据集，通过将历史负荷数据等效为时间序列，而实现对历史负荷数据的数据挖掘和模型预测。然而实际上随着电力大数据概念的形成，可以更多的从影响负荷的各种因素出发进行负荷预测。在文献[52]中提出实际上预测模型的输入数据可以变历史负荷数据集为影响负荷的各种因素情况，例如可以涵盖日期属性（如月份类型、星期类型、节假日类型等）、天气数据（如温度、湿度、光照强度等），甚至在电力市场背景下还应涵盖需求侧管理信息与实时电价情况，通过这些影响负荷的因素与负荷数据之间进行学习和预测，通过丰富数据集中的数据类型情况对于进一步提升模型预测性能具有重要意义。

（2） 预测模型的计算硬件部署

在本文中，由于受制于计算机硬件条件限制，所有的短期负荷预测模型训练全部都是通过计算机CPU运算实现。然而实际上目前主流的深度学习模型训练硬件已经广泛应用GPU(Graphics Processing Unit)为主要运算设备。GPU实际上属于一种大规模并行计算架构，其拥有数以千计的专门用于计算密集型任务的更小、更高效核心，因此模型训练这一类需要大规模迭代计算的领域具有远胜于CPU的处理效率。据GPU厂商NVIDIA的在机器学习应用的基准测算[53]，使用GPU加速后执行机器学习任务的速度比CPU提升33倍。对于电力大数据背景，这一硬件突破同样可以极大地缩短短期负荷预测模型的训练时间，进而提升算法性能。此外由于GPU的高并行运算速度，将可能对本文第4章中提出的分布式集群参数服务器架构的集群通信产生巨大压力，这也是当今分布式深度学习系统领域的研究热点问题之一，因此未来需要在有相关设备的情况下对GPU硬件进行模型实现，并进一步改进预测模型的分布式并行计算架构，以适用于未来电力系统实际工作中的需求。

总之，作为四川大学电气工程及其自动化专业的本科毕业设计，本文从现实生产中存在的复杂工程问题出发，根据目前智能电网的发展趋势较为前瞻性地研究本课题，并且运用数据科学、计算机科学和人工智能领域最新的理论成果和程序平台架构，探究实现深度学习模型与分布式深度学习系统在电力数据挖掘领域的应用，设计提出了相应的具有应对海量数据处理能力的高精度预测模型和高性能分布式并行算法，具备较高的模型和算法创新性。通过综合应用本科四年期间的电力系统理论、数学建模思维和计算机应用能力，从一定程度上对这一个复杂工程问题给予创新性解决方案。为了验证模型和算法性能，本文所有数据集均采用真实电力负荷数据，且应用于模型对照的算法也均为过去和目前电力系统常用的经典短期负荷预测算法，从而锻炼了工程意识和解决实际工程问题的能力，也让本文所提出的模型和算法更具有实践意义。大量跨学科理论和模型的引入更锻炼和培养了我的中英文文献检索能力与终生学习能力，并为下一步研究生阶段的学术深造打下了坚实的基础，本文对这一领域未来学术研究具有一定的参考价值和实践意义。

参考文献

[1] D. W. Bunn and E. D. Farmer, Comparative Models for Electrical Load Forecasting, 1st ed. New York, NY, USA: Wiley, 1985:41-43.

[2]康重庆，夏清，张伯明.电力系统负荷预测研究综述与发展方向的探讨[J].电力系统自动化，2004，28(13):1-5.

[3]王德文，孙志伟.电力用户侧大数据分析与并行负荷预测[J].中国电机工程学报，2015(35):528-536.

[4]孙志军，薛磊，许阳明，王正.深度学习研究综述[J].计算机应用研究，2012，29(5):2809.

[5]姚建国，杨胜春，王珂，杨争林，宋晓芳.智能电网“源—网—荷”互动运行控制概念及研究框架[J].电力系统自动化，2012,36(11):1-5.

[6]梁煜. 基于云平台的重卡车联网运营服务系统的分析与设计[D].厦门大学软件工程硕士毕业论文，2016.

[7]王松桂，陈敏，陈立萍.线性统计模型[M].北京:高等教育出版社，1999.

[8]方开泰，全辉.实用回归分析[M].北京:科学出版社，1988.

[9]侯志俭，吴际瞬，张绮雨等.电力系统短期负荷预报的几种改进手段[J].电力系统自动化，1996，20(7):27-31.

[10]莫维仁，张伯明，孙宏斌，胡子珩.短期负荷预测中选择相似日的探讨[J].清华大学学报（自然科学版），2004，44(1):106-109.

[11]黎灿兵，李晓辉，赵瑞，李金龙，刘晓光.电力短期负荷预测相似日选取算法[J].电力系统自动化，32(9)：69-72.

[12]刘思峰，郭天榜.灰色系统理论及其应用[M].北京:科学出版社，1999.

[13]李鹰，赵振江.灰色模型在普通日短期电力负荷预测中的应用[J].长沙电力学院学报，2003，18(1):15—17.

[14]李鹰，卢炎生.灰色模型GM(I，1)及其改进模型在短期特殊日电力负荷预测中的应用[J].桂林工学院学报，2002，22(4):418—420.

[15]邰能灵，侯志俭.小波模糊神经网络在电力系统短期负荷预测中的应用[J].中国电机工程学报，2004，24(1):24-27.

[16]严华，吴捷，马志强，吴列鑫.模糊集理论在电力系统短期负荷预测中的应用[J].电力系统自动化，2000，24(11):68-69.

[17]张国江，邱家驹，李继红.基于模糊推理系统的多因素电力负荷预测[J].电力系统自动化，2002，26(5):1-4.

[18]吴捷，严华.基于自适应最优模糊逻辑系统的短期负荷预测方法[J].电力系统自动化，1999，23(17):2-7.

[19]罗玮，严正.基于广义学习矢量量化和支持向量机的混合短期负荷预测方法[J].电网技术，2008，32(13):62-65.

[20]李云飞.支持向量机在电力系统短期负荷预测中的应用及改进[D].西南交通大学电力系统及其自动化硕士学位论文，2006.

[21]李元诚，方廷健，于尔铿.短期负荷预测的支持向量机方法研究[J].中国电机工程学报，2003，23(6):55-59.
[22]周佃民，管晓宏，孙婕，黄勇.基于神经网络的电力系统短期负荷预测研究[J].电网技术，2002，26(2):10-17.
[23]陈耀武，汪乐宇，龙洪玉.基于组合式神经网络的短期电力负荷预测模型[J].中国电机工程学报，2001，21(4):79-81.
[24]雷绍兰，孙才新，周湶，张晓星，程其云.基于径向基神经网络和自适应神经模糊系统的电力短期负荷预测方法[J].中国电机工程学报，2005，25(22):78-82.
[25]高山，单渊达.神经网络短期负荷预测输入变量选择新方法[J].电力系统自动化，2001，25(22):1-6.
[26]丁坚勇，刘云.基于负荷特征提取的神经网络短期负荷预测[J].高电压技术，2004，30(12):47-49.
[27]师彪，李郁侠，于新花，闫旺.基于改进粒子群-模糊神经网络的短期电力负荷预测[J].系统工程理论与实践，2010，30(1):158-166.
[28]葛少云，贾鸥莎，刘洪.基于遗传灰色神经网络模型的实时电价条件下短期电力负荷预测[J].电网技术，2012，36(1):224-227.
[29]丁军威，孙雅明.基于混沌学习算法的神经网络短期负荷预测[J].电力系统自动化，2000，24(2)：33-35.
[30]祝燕萍，方鸽飞.基于动态自适应神经网络和人体舒适度的短期负荷预测[J].电力系统保护与控制，2012，40(1):58-61.
[31] G. Box and G. Jenkins, Time Series Analysis: Forecasting and Control, Rev. ed. Oakland, CA, USA: Holden-Day, Cop., 1976.
[32] M. Cho, J. Hwang, and C. Chen, “Customer short term load fore-casting by using ARIMA transfer function model,” in Proc. Int. Conf. Energy Manage. Power Del., 1995, vol. 1, pp. 317–322.
[33] N. Amjady, “Short-term hourly load forecasting using time-series mod- eling with peak load estimation capability,” IEEE Trans. Power Syst., vol. 16, no. 4, pp. 798–805, Aug. 2001.
[34] S.-J. Huang, S. Member, and K.-R. Shih, “Short-term load forecast- ing va ARMA model identification including non-Gaussian,” IEEE Trans. Power Syst., vol. 18, no. 2, pp. 673–679, May 2003.
[35] H. Hippert, C. Pedreira, and R. Souza, “Neural networks for short-term load forecasting: A review and evaluation,” IEEE Trans. Power Syst., vol. 16, no. 1, pp. 44–55, Feb. 2001.
[36] A. S. Khwaja, M. Naeem, A. Anpalagan, A. Venetsanopoulos, and B. Venkatesh, “Improved short-term load forecasting using bagged neural networks,” Electr. Power Syst. Res., vol. 125, pp. 109–115, 2015.
[37] S. Kouhi and F. Keynia, “A new cascade NN based method to short-term load forecast in deregulated electricity market,” Energy Convers. Manage., vol. 71, pp. 76–83, 2013.

[38] C. Cecati, J. Kolbusz, P. Siano, and B. M. Wilamowski, “A novel RBF training algorithm for short-term electric load forecasting and comparative studies,” IEEE Trans. Ind. Electron., vol. 62, no. 10, pp. 6519–6529, Oct. 2015.

[39] N.-S.Pang and Y.-l.Shi,“Research on short-term load forecasting based on adaptive hybrid genetic optimization BP neural network algorithm,” in Proc. Int. Conf. Manage. Sci. Eng. 15th Annu. Conf. Proc., Long Beach, CA, USA, pp. 1563–1568, 2008.

[40] S.H.Ling, F.H.F.Leung, H.K.Lam, Y.S.Lee, and P.K.S.Tam,“Anovel genetic-algorithm-based neural network for short-term load forecasting,” IEEE Trans. Ind. Electron., vol. 50, no. 4, pp. 793–799, Aug. 2003.

[41] S. H. Ling, F. H. F. Leung, H. K. Lam, and P. K. S. Tam, “Short- term electric load forecasting based on a neural fuzzy network,” IEEE Trans. Ind. Electron., vol. 50, no. 6, pp. 1305–1316, Dec. 2003.

[42] Warren S.McCulloch, and Walter Pitts, “A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biology, vol. 52, no. 1/2, pp. 99-115, 1990.

[43] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, “Learning representations by back-propagating errors”, Nature, vol. 323, no. 9, pp. 533–536, Oct. 1986.

[44]刘知青，吴修竹.解读AlphaGo背后的人工智能技术[J].控制理论与应用，2016，33(12):1685-1687.

[45] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961.

[46] Sepp Hochreiter, Jürgen Schmidhuber. "Long short-term memory". Neural Computation. vol. 9, no. 8, pp. 1735–1780,1997.

[47] Felix A. Gers, Jürgen Schmidhuber, Fred Cummins. "Learning to Forget: Continual Prediction with LSTM". Neural Computation. vol. 12, no. 10, pp. 2451–2471,2000.

[48] Hinton, Geoffrey E., Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, Salakhutdinov, Ruslan R.. "Improving neural networks by preventing co-adaptation of feature detectors" ,2012.

[49] Dean, Jeff, Monga, Rajat, et al. . "TensorFlow: Large-scale machine learning on heterogeneous systems". TensorFlow.org. Google Research. Retrieved November 10, 2015.

[50] Zaharia, Matei, Chowdhury, Mosharaf, Franklin, Michael J., Shenker, Scott; Stoica, Ion. Spark: Cluster Computing with Working Sets. USENIX Workshop on Hot Topics in Cloud Computing (HotCloud), 2010.

[51]薛禹胜，赖业宁.大能源思维与大数据思维的融合（一）大数据与电力大数据[J].电力系统自动化，2016，40(1):1-6.

[52]孔祥玉， 郑锋， 鄂志君， 曹旌， 王鑫.基于深度信念网络的短期负荷预测方法[J].电力系统自动化，2018，42(5):133-138.

[53] Lindholm E, Nickolls J, Oberman S, Montrym J. NVIDIA Tesla: A Unified Graphics and Computing Architecture. IEEE Micro 28, pp. 39–55, 2008.

致 谢

时光荏苒，岁月如梭，转眼间自己四年的本科学习生涯即将结束，值此论文完成之际，谨向关心、指导和帮助我的老师、同学与家人们表示衷心的感谢。

本文的完成得到了多方面的帮助和支持，在此要特别感谢论文指导老师刘洋老师对我的教育和指导。在本科四年的学习期间，刘老师教授我多门专业核心课程，并在课后有诸多交流，培养起我对专业的兴趣，并且指导我积极探索科研。在本学期进行本毕业设计期间，刘老师严谨的治学态度、渊博的学术知识、勤奋的工作状态都给我留下了深刻印象，让我十分钦佩。老师对我的教导和有影响必将在我今后的深造和工作中起到极大的作用。

在本科学习生涯中，还有幸得到了电气信息学院与吴玉章学院诸多老师的关怀和指导，他们的高尚师德和渊博学识指引着我从入校伊始的大学新生一步步成长为一名合格的本科毕业生，并培养起我解决实际复杂工程问题的能力与终生学习的意识，这一切都将成为我一生的财富。

其次要感谢史宇昊、张锐、刘昆鹏、刘书瀚、郭人旭、龚翼、李泽明等同窗好友，在四年的本科学习生涯期间共同生活和进步，无论在学业还是生活上，同窗好友们都曾给予我很大的支持和帮助，让我在大学期间的四年生活殊为难忘。

同时要感谢我亲爱的父母，正是你们的多年辛勤付出和默默支持让我得以在美丽的大学校园不断成长和前进，尽管家境并不富裕，但父母总是默默操劳给我提供安心的读书和学术环境，你们的敬业精神与生活态度给予了我无尽的勇气和力量，更是我今后人生路上的榜样。

最后要感谢母校四川大学对我的培养和支持，让我有幸在本科期间收获了无尽的学识和成长，并且有机会在本科期间两次公费赴美，在宾夕法尼亚大学和加州大学洛杉矶分校锻炼我的国际视野和学术科研能力，并且有幸在未来研究生阶段的深造中继续自己的科研求索。我将永远铭记“海纳百川，有容乃大”的校训，在未来的人生道路上永远积极进取，勇攀高峰！

尹思凯

2018 年 5 月 18 日于四川大学