电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 **基于神经网络的多变量时间序列预测**

专业学位类别	工程硕士
学　　号	**201922080611**
作者姓名	陈超
指导教师	高辉　　教　授
学　　院	计算机科学与工程学院

分类号 TP183 密级 公开

UDC [注1] 004.8

学 位 论 文

基于神经网络的多变量时间序列预测

（题名和副题名）

陈超

（作者姓名）

指导教师 高辉 教 授

电子科技大学 成 都

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工程硕士

专业学位领域 计算机技术

提交论文日期 2022 年 3 月 28 日 论文答辩日期 2022 年 5 月 31 日

学位授予单位和日期 电子科技大学 2022 年 6 月

答辩委员会主席 李建平

评阅人

注 1：注明《国际十进分类法 UDC》的类号。

Multivariate Time Prediction Based on Neural Network

A Master Thesis Submitted to
University of Electronic Science and Technology of China

Discipline	**Master of Engineering**
Student ID	**201922080611**
Author	**Chao Chen**
Supervisor	**Prof. Hui Gao**
School	**School of Computer Science and Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：陈超　　日期：2022 年 6 月 1 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将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、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：陈超　　导师签名：

日期：2022 年 6 月 1 日

摘 要

随着云计算和大数据的不断发展，人们能够计算和存储的数据越来越多。人们希望能从大量相似的历史时间序列数据中预测未来、辅助决策。因此，短期的时间序列预测问题是一个非常有意义的研究课题，其成果可以运用于商品库存周转，交通流量预警以及金融投资等领域。本文借助深度学习的相关技术与理论，在单步预测、多步预测两种不同的预测方法开展研究。本文主要工作包括：

（1）针对多变量单步预测问题，本文提出了 GCNN-DeepAR 模型，该模型由多核门控卷积神经网络和 LSTM 长短记忆神经网络组成，通过混合两个神经网络，加强模型对短周期和长周期的时序模式捕获。模型先对时间序列数据进行概率分布假设，然后通过概率分布损失函数训练学习到时间序列概率分布中的参数集合，最后通过对该分布进行采样生成单步的预测值，并给出其预测的上下界。在两个数据集合上的对比结果表明，GCNN-DeepAR 模型取得比较好的结果。

（2）传统的方法通过迭代单步预测生成多步预测，这样会导致累计误差影响精度，单步迭代方法只适用于单变量时间序列预测，不适用于多变量时间序列预测。针对多变量多步预测问题，本文提出了能直接预测多步的 Transformer-AR 模型，该模型由多核门控卷积神经网络和 Transformer 神经网络编码器组成，通过混合两个神经网络，加强模型对短周期和长周期的时序模式捕获。针对 Transformer 神经网络的注意力机制是全局的、双向的，不满足我们时间序列的时间依赖性，本文设计了一种掩码方式，将 Transformer 的编码器部分应用到多步时间序列预测问题中。该模型使用分位数损失函数进行训练学习，对每个预测时间点同时给出 $\rho_{10} = 0.1$，$\rho_{50} = 0.5$，$\rho_{90} = 0.9$ 分位数的预测值，量化预测的不确定性。在两个真实的数据集合上的对比结果表明，Transformer-AR 取得了比较好的结果。

关键词：时间序列预测，单步概率预测，多步概率预测，深度学习

ABSTRACT

With the continuous development of cloud computing and big data, people can calculate and store more and more data. People hope to predict the future and assist decision-making from a large amount of similar historical time series data. Therefore, the short-term time series forecasting problem is a very meaningful research topic, and its results can be used in the fields of commodity inventory turnover, traffic flow warning and financial investment. With the deep learning technologies and theories, this thesis conducts research on two different forecasting methods: single-step forecasting and multi-step forecasting. The main work of this thesis includes:

(1) For the multivariate single-step prediction problem, this thesis proposes the GCNN-DeepAR model, which consists of The multi-kernel gated convolutional neural network and the LSTM long-short-term memory neural network, and by mixing the two neural networks, the model can enhance the capture of short-period and long-period temporal patterns. The model first makes probability distribution assumptions on the time series data, then trains the parameter set in the time series probability distribution through the probability distribution loss function, and finally generates a single-step predicted value by sampling the distribution, and gives the upper and lower prediction boundary. The comparison results on the two datasets show that the GCNN-DeepAR model achieves better results.

(2) The traditional method generates multi-step forecast by iterative single-step forecast, which will lead to accumulative error affecting the accuracy. one-step iteratation method is only applicable to univariate time series prediction, not multivariate time series prediction. For the multivariable multi-step prediction problem, this thesis proposes a Transformer-AR model that can directly predict multi-step. This model consists of a multi-kernel gated convolutional nerual network and a Transformer encoder, and by mixing the two neural networks, the model can enhance the capture of short-period and long-period temporal patterns. For the attention mechanism of the Transformer nerual network is global and bidirectional, which does not satisfy the time dependence of time series, this thesis designs a mask to apply the Transformer encoder to the multi-step series prediction problem. This model is trained using the quantile loss function, which gives$\rho_{10} = 0.1, \rho_{50} = 0.5, \rho_{90} = 0.9$ quantile predicted values, quantifying the uncer-

tainty of the prediction. The comparison results on two real datasets show that TransformerAR achieves better results.

Keywords: Time series forecasting, single-step probabilistic forecasting,multi-step probabilistic forecasting, deep learning

目 录

第一章 绪 论.....1
1.1 研究工作的背景与意义.....1
1.2 时间序列预测的国内外研究历史与现状.....2
1.2.1 基于统计学的时间序列预测.....2
1.2.2 基于机器学习的时间序列预测.....3
1.2.3 基于深度学习的时间序列预测.....4
1.3 本文的主要贡献与创新.....5
1.4 本论文的结构安排.....6
第二章 时间序列预测相关理论与技术.....8
2.1 时间序列概述.....8
2.2 时间序列数据处理.....9
2.2.1 时间序列数据划分.....9
2.2.2 时序数据的预处理.....10
2.3 深度学习相关理论.....12
2.3.1 前馈神经网络.....13
2.3.2 卷积神经网络.....14
2.3.3 循环神经网络.....18
2.3.4 序列到序列神经网络.....21
2.4 本章小结.....25
第三章 基于神经网络的多变量单步概率预测模型.....26
3.1 多变量单步预测问题描述.....26
3.2 GCNN-DeepAR 概率预测模型.....27
3.2.1 输入预处理.....27
3.2.2 一维门控卷积.....29
3.2.3 多层 LSTM 循环神经网络.....31
3.2.4 时间独立分布参数层.....32
3.2.5 训练和推理过程.....33
3.3 实验过程与结果.....33
3.3.1 实验内容介绍.....33
3.3.2 实验数据集分析与处理.....37

3.3.3 对比实验组 38
3.3.4 预测效果分析 39
3.3.5 消融实验 42
3.4 本章小结 43
第四章 基于神经网络的多变量多步概率预测模型 44
4.1 多变量多步概率预测问题描述 44
4.2 Transformer-AR 多步预测模型描述 46
4.2.1 输入预处理 47
4.2.2 一维门控卷积层 49
4.2.3 Masked Transformer 编码器 50
4.2.4 三步时间独立线性层 52
4.2.5 训练和推理过程 53
4.3 实验过程与结果 55
4.3.1 实验内容介绍 55
4.3.2 数据集分析与处理 57
4.3.3 对比实验组 58
4.3.4 预测效果分析 59
4.3.5 消融实验 63
4.4 本章小结 63
第五章 全文总结与展望 65
5.1 全文总结 66
5.2 后续工作展望 66
致 谢 67
参考文献 68

第一章 绪 论

1.1 研究工作的背景与意义

时间序列预测，是利用历史上观测到的时间序列数据和该问题领域下掌握的先验知识，来挖掘和分析时间序列的周期性、波动性和趋势性等各种时间序列的模式，进而使用掌握的模式来预测未来时刻发生的结果。在当今时代中，时间序列的挖掘和分析预测已经成为许多领域的重要和基本的方法工具，可以依据获得的预测结果进行资源优化分配、对未来趋势进行预判等。比如牛奶加工厂商通过对奶农提供的原奶的生化指标进行时间序列预测，预测未来几天的原奶供应的品质，帮助牛奶加工企业提供原奶供应品质预警，进行有效的供应原材料管理；电商公司对商品的销售数量进行时间序列预测分析，预测未来某段时间商品的销售量，提前进行公司库存的优化，促进公司的物流流转效率；智慧零售小店，追踪每个商品的销售时序数据，通过短期时序预测模型，优化店内每种商品的库存和上架率，降低小店的运营成本，提高小店的收入。

为了建立准确和高效的时间序列模型，长期以来有许多的研究人员进行该项研究，提出了统计学经典的 ARIMA[1] 等方法。传统的时间序列研究主要是以统计方法和随机过程为基础，对滞后的数据进行加权平均计算、对滞后的白噪声也进行加权平均计算，然后再将这两项结果进行相加，最终作为未来一时刻的预测结果，其预测残差还检验是否满足高斯分布。传统的时间序列预测模型虽然简单，然而仅仅从时间的维度来预测未来的值，没有考虑到相关的先验知识和其他的特征，且主要是对线性关系进行建模，只能适应平稳的时间序列数据，对应日常生活中的非平稳的时间序列问题，其预测的精度会大大降低。

为了解决传统的时间序列模型的缺点，近几年随着机器学习和深度学习等方法的发展，机器学习和深度学习等方法逐渐被引入到时间序列领域中，用于解决非线性、非平稳的时间序列问题。机器学习中对影响时间序列中的特征进行处理，通过使用滑动窗口来构造数据集，切分训练集和测试集。手动特征选择和构造，数据清理等方法对原始的时间序列进行预处理，然后再通过选择相关的机器学习模型进行训练和评估，优化模型的泛化能力。到如今，工业界大规模的机器学习模型，诸如 XGboost[2](eXtreme Gradient Boosting)、LightGBM[3](Gradient Boosting Decision Tree)、CatBoost[4,5]、人工神经网络[6](Arificial Neural Networks，简写为 ANN) 等机器学习模型已经被广泛运用到时间序列预测中。相比与传统的时间序列预测方法，机器学习可以添加各种与预测结果有因果关系的特征组合，能够建

立足够复杂的预测模型，捕获非线性的预测模式，适应非平稳的数据序列数据，提高预测的精确度，但是其难以挖掘时间依赖性和特征间的相关性，难以处理大规模相似时间依赖的时间序列模型。

最近几年，随着计算机硬件，比如 GPU、FPGA 等的算力性能不断的提高；深度学习软件，比如 Tensorflow[7]、Torch[8] 等深度学习框架易用性的提高；大数据的搜集和存储等能力的提高。深度学习发展迅速，已被许多学者运用到时间序列挖掘和预测问题中，包括用于处理自然语言的 Transformer 神经网络和循环神经网络，用于处理图像的卷积神经网络。深度神经网络中，随着网络深度的提升，神经网络拥有更多的参数，神经网络更能拟合复杂的函数曲线，同时深度神经网络还拥有更好的特征提取能力，善于提取数据中的特征，比如在图像处理领域中，使用多层卷积神经来提取图片中的特征，相比原始的图像 HOG，SIFT 等特征提取方法，有了很大的提升。不再需要人工的特征处理，能够通过任务自动地提取相关的特征。使得深度学习方法对传统的机器学习方法对大量数据的处理力更强，不需要复杂的人工特征工程，更少的超参数调优。除此之外，卷积神经网络和循环神经网络，能捕获时间序列数据中的数据时间依赖关系，使得深度学习方法的预测准确率得到了进一步的提升。

综上所述，深度学习方法能够更好地提取数据中的特征，能够学习长期和短期的时间依赖，能更好的挖掘数据中的隐性关系，大幅提升预测的精确率和更好的模型泛化能力。

1.2 时间序列预测的国内外研究历史与现状

随着统计学和机器学习的发展，时间序列预测主要分为三种类型，接下来本章将详细阐述这三类方法在国内外的研究现状和成果。

1.2.1 基于统计学的时间序列预测

基于统计学的时间序列预测方法早期主要是研究平稳的线性的时间序列关系，之后经过长期的发展，从只能通过线性的、平稳的、单变量的简单时间序列建模逐步发展成非线性的、非平稳的、多元的复杂时间序列建模。

基于统计学的预测方法最早是在 1927 年由 Yule、Walker 等人提出的，文中提出了一种线性时间序列预测模型，称为自回归（AutoRegressive，简称为 AR）模型，AR 模型通过时间序列的历史线性组合加上当前的白噪声来预测未来时刻点的值。随后 Slutzky 于 1937 年提出了移动平均（Moving Average，简称为 MA）模型，MA 模型通过线性组合当前和历史上的白噪声来预测当前时点，每个时间点上的

白噪声假设是独立同分布，通常来自均值为 0、方差为 1 的高斯分布。它于 AR 模型的区别主要是两方面，第一、白噪声 ε_{t-1} 直接影响当前时点预测；第二、只有有限个历史时刻的白噪声影响当前时点预测。Wold 学者将 MA 模型和 AR 模型进行组合，提出了自回归移动平均（ARMA）模型。MA、AR 和 ARMA 这三种模型成为经典的时间序列预测的方法，适用于平稳的、单变量的时间序列预测问题。为了解决非平稳时间序列预测问题，Box G E P, Jenkins G M.(1968)[1] 学者利用一阶差分，二阶差分，多阶差分运算，将非平稳的序列转化为平稳序列的思想（一般只是去除趋势项，让均值平稳。但是对非平稳的方差/自方差，差分运算无法让其平稳化），提出了差分移动平均自回归（AutoRegressive Integrated Moving Average，简称 ARIMA）模型。ARIMA 模型的提出引发了巨大的反响，其能适应一部分非平稳的时间序列预测，优秀理论算法，极大的促进了时间序列预测在各个领域的广泛运用。另外一方面，向量自回归（VAR）由于其简单性，可以说是多变量时间序列预测中使用最广泛的模型。VAR 模型自然地将 AR 模型扩展到多变量预测，忽略输出变量间的依赖关系。VAR 的模型容量随着时间窗口大小呈线性增加，随着变量数据呈二次增长。在处理长距离时间序列模式，模型很容易过拟合。

1.2.2 基于机器学习的时间序列预测

基于统计学的预测方法在使用时具有极大的局限性，比如差分预处理数据后，需要检验其是否满足平稳性要求，需要判定 p 阶数和 q 阶数，最后还要进行模型检验，判定残差是否满足正态分布。因此，可以看出 ARIMA 模型，人工特征工程工作量大，并且模型主要是捕获线性关系，只适用于单变量的时间序列预测问题，达不到理想的预测精度。随着机器学习的相关理论和技术的不断发展，机器学习的方法和模型也开始应用到时间序列预测问题。

时间序列预测问题也可以被视为具有时变参数的标准回归问题。基于机器学习的预测方法主要包含 SVM[9,10]（Support Vector Machine）、GBDT[11]（Gradient Boosting Decision Tree）、线性回归[12]（Linear Regression）、高斯过程[13]（GP）等方法。通过特征工程构造对预测结果具有因果的特征集合，构造相应的训练数据集和验证数据集，使用合适的损失函数并训练模型，在验证数据集上进行超参数调优，加强模型的泛化能力，最终得到时序预测模型。

SVM 模型通过使用核函数，隐式地使数据从低维空间映射到高维空间，通过超平面线性划分数据，最终实现分类。SVM 主要是运用到分类问题，为了让 SVM 运用到回归问题上，学者基于 SVM 的思想提出了支持向量回归[14]（Support Vector Regression，简称为 SVR）模型，将 SVM 运用到了回归问题上。SVR 模型

也能运用到时间序列预测的领域内，例如，由张朝阳等[15]人提出使用基于最小二乘支持向量机方法来预测交通流量时间序列，相比与传统的时间序列预测方法，取得较好的预测效果。

GBDT 模型是一种集成学习的方法，使用决策回归树[16]作为一个弱学习器，通过训练多颗决策回归树来拟合模型预测的残差，然后集成所有的决策树的预测结果进行预测，具有较强的学习能力和泛化能力。主要包含 XGboost、LightGBM、CatBoost 等主流集成学习模型。决策树也可以运用到时间序列预测的领域中，例如，由孙晓黎等[17]人提出基于 XGboost 的轨道交通短时客流预测精度分析，来进行轨道客流预测，相比与其他方法，基于 XGboost 的决策树模型实现了高精度的预测。

高斯过程（GP）是一种用于对函数连续域上的分布进行建模的非参数方法。这与由 VAR 和 SVR 等参数化函数类定义的模型形成对比。高斯过程（GP）可以应用到多变量时间序列预测任务，并且可以用作贝叶斯推理中函数空间的先验。例如，由 Frigola R, Lindsten F, Schön T B, et al.[18]提出了一种具有 GP 先验的完全贝叶斯方法，用于非线性状态空间模型，能够捕捉复杂的动态现象。

1.2.3 基于深度学习的时间序列预测

虽然基于机器学习的时间序列预测的方法比基于统计学的时间序列预测方法要好，比如适用多变量时间序列数据、更少的人工特征工程、良好的预测能力和非线性的拟合能力。然而机器学习方法只考虑当前时间的特征集合，没法建模表现时间序列数据中时间的依赖性，同时当数据量变得很多时，机器学习的学习能力不能随着数据增多而增强，无法获得准确较高的预测结果，为了解决基于机器学习的预测方法存在的各种问题，研究者开始使用深度学习来解决机器学习在时间序列预测中所遇到的问题。

机器学习方法，主要是依赖人工特征工程，而人工特征工程很大一部分决定预测的准确率，然而深度学习主要是由神经网络进行特征的提取，无需人工选择特征。深度学习的神经网络可以无限的深，拥有大数据级的参数，具有更好的学习和泛化能力。深度学习在语音领域、自然语言、图像处理等领域的优秀表现，说明了深度神经网络也能处理多维数据、非线性模式、非平稳数据的问题，也就意味着在时间序列预测领域有着更好的表现。

卷积神经网络[19]（Convolutional Neural Network，简称 CNN），主要通过卷积、池化等方法进行特征的提取，并被广泛运用在图像、语音、自然语言处理等领域中。卷积操作具有一定的视野范围，可以用来短距离的时间依赖建模，因此可以

运用到时间序列预测中。近年来，一维卷积的 CNN 模型被应用到时间序列预测问题中，例如，裴艳宇等[20]人提出的一维卷积神经网络特征提取下微震能级时序预测，使用一维卷积神经网络对矿山微震能量时序数据进行预测，提高了微震能级时序预测的正确率。

循环神经网络[21]（Recurrent Neural Network，简称 RNN），作为一种循环的神经网络，在自然语言处理领域证明了其有捕获长时间依赖的能力，因此可以被运用到时间序列预测问题中。但是对长时间的预测问题，RNN 训练时容易出现梯度爆炸或梯度消失的问题，使得模型难以训练和收敛。因此学者提出了长短记忆神经网络（LSTM），通过门控操作来控制单元状态的记忆和遗忘，记住需要模型需要的长时间记忆的信息，忘记不重要的历史信息。在一定程度上解决了梯度爆炸或梯度消失的问题。例如，陈亿雄等[22]人使用长短记忆神经网络（Long Short-Term Memory）预测未来每周的流行性感冒爆发趋势的可行性，实验证明在数据量大和非平稳及周期特征的情况下，LSTM 神经网络的拟合更接近真实值，证明了 LSTM 网络在时间序列预测中的有效性。除此之外，也有学者提出了更简化的循环神经网络的变种门控循环神经网络（GRU），通过将输入门控单元和遗忘门控单元合并为更新门控单元，减少了 LSTM 网络的参数量，实现更快的训练和推理。

Transformer 神经网络[23]，只使用多头自注意力和前馈神经网络，在自然处理领域获得巨大的成功，同时在图像处理领域也获得成功，因此有研究者也将 Transformer 神经网络运用到时间序列预测问题中。相比于循环神经网络来说，Transformer 神经网络完全依赖于自注意力机制，不需要像循环神经网络一样迭代输入。因此 Transformer 拥有更大的并行度，计算量比循环神经网络要小，不像循环神经网络容易出现梯度爆炸或梯度消失，更能处理长序列和大数据集。例如，Lim B et al. (2021)[24] 提出了基于 Transformer 神经网络架构的 TFT 时间序列预测模型，对电力消耗时间序列进行预测，得到了很好的预测结果，证明了 Transformer 神经网络在时间序列预测中的有效性。

1.3 本文的主要贡献与创新

本文旨在研究利用深度神经网络模型来解决时间序列的预测问题，通过使用参数估计损失函数（如高斯分布损失函数）和非参数估计损失函数（如分位数损失函数）对时间序列进行概率预测建模，给出预测的上下界，量化模型预测不确定性的。

论文的主要工作如下：

针对多变量单步时间序列预测问题，我们提出了 GCNN-DeepAR 多变量单步

概率预测模型，并详细介绍该模型。对 DeepAR 模型进行改进，增加了多核一维门控卷积神经网络和时间独立分布参数层神经网络。该模型利用不同核函数的一维门控卷积神经网络，对多变量特征进行不同的短周期信息的捕获和特征融合，然后利用该融合的特征数据，使用 LSTM 循环神经网络进行长周期时间相关性挖掘，最后对原始的时间序列进行概率分布假设，使用高斯分布损失函数对模型进行训练，实现单步概率预测。实验对比验证 GCNN-DeepAR 模型在单步时间序列预测问题上的有效性和准确性。

针对多变量多步时间序列预测问题，我们提出了 Transformer-AR 多变量多步概率预测模型，并详细介绍该模型。改进了时间序列数据预处理的尺度变化公式和反变化公式，让其适应有正有负的数据特征。该模型利用了不同核函数的卷积神经网络，对多变量特征进行不同的短周期信息的捕获和特征融合，然后利用该融合的特征数据，使用 Masked Transformer 神经网络编码器对时间序列数据进行长周期模式挖掘，最后使用分位数损失函数对模型进行训练，实现多步概率预测。通过实验对比验证 Transformer-AR 模型在多步时间序列预测问题上的有效性和准确性。

1.4 本论文的结构安排

本文的章节结构安排如下：

第一章，绪论，概述时间序列预测问题的研究意义和背景，总结时间序列预测方法的国内外研究现状和成果，最后阐述本文的研究内容和组织结构。

第二章，时间序列预测相关的研究的理论与技术，阐述时间序列预测的相关概念。阐述时间序列中的数据预处理方法，如滑动窗口生成数据、数据的重采样、数据的变换和反变换等，并简要介绍深度学习相关的理论，如前馈神经网络、卷积神经网络、循环神经网络、序列到序列神经网络等。

第三章，针对多变量单步时间序列预测问题，我们提出了一种 GCNN-DeepAR 概率预测模型，并详细介绍该模型。对原始的时间序列数据进行滑动窗口切分，构造相应的训练数据集和测试数据集，将单步预测问题转换为自监督学习。对真实时间序列数据进行预处理、尺度变换，在两个真实的数据集上进行对比实验，验证我们的 GCNN-DeepAR 模型的单步概率预测能力。

第四章，针对多变量多步时间序列预测问题，我们提出了一种基于 Transformer-AR 神经网络预测模型，并详细介绍该模型。对原始的时间序列数据进行滑动窗口切分，构造相应的训练数据集和测试数据集，将多步预测问题转换为自监督学习。构造适合模型的输入数据格式和掩码设计，对真实的时间序列数据进行预处

理、尺度变换，在两个真实的数据集上进行对比实验，验证我们的 Transformer-AR 模型的多步概率预测能力。

第五章，总结与展望，概况本文的研究内容，对时间序列预测方法未来的研究方向进行展望。

第二章 时间序列预测相关理论与技术

本章阐述基于深度学习的时间序列预测方法中所用的技术和深度学习相关理论。具体包括如下三个部分：时间序列概述、时间序列数据处理、深度神经网络理论。其中时间序列概述将阐述时间序列的特点和时间序列预测问题的分类；时间序列数据处理将阐述数据的切分、预处理、归一化等问题；深度学习相关理论将阐述深度学习中的前馈神经网络、卷积神经网络、长短记忆神经网络、序列到序列神经网络等内容。

2.1 时间序列概述

时间序列数据存在于现实生活中的各个方面，比如金融对冲领域、自动驾驶领域、农业生产领域、交通领域等。基于现存的历史时间序列数据，挖掘和学习数据中的规律（如周期性、趋势性、波动性），并建立适应的时间序列预测模型，利用该模型进行时间序列预测，为人类的生活和决策提供一种参考信息，辅助人们更好地掌握生产活动。

时间序列通常使用四个部分进行描述。趋势项，一个平滑没有上下波动长期的曲线，主要有增长、下降或者稳定趋势，例如近年来随着时间增长，我国的 GDP 总量持续增长。周期项，是一个跟随期间所变化的变动，以年、月、季度或者固定时间单位为周期波动，比如道路交通量随着时间的变化而变化，上下班时间车流量高，其余时间车流量少。循环波动波动，没有固定周期的波动，通常其平均周期比较长，例如印度的 CPI（consumer price index）指数，大概是 2-2.5 年，但是在 1990 年前的波动周期比 1990 年后的波动周期大。其它变动项，通常我们假设是随机变量，比如白噪声。

根据预测的步数可以把时间序列预测问题分为，单步时间序列预测问题和多步时间序列预测问题。单步预测方法是使用过去历史的观察数据来预测下一时间点的数据，通过不断多步迭代预测来获得更多的预测结果。而多步预测方法是利用过去观察数据直接生成多步预测结果。假设我们采样了 T 个时间序列样本，$\boldsymbol{h}_{1:\mathrm{T}} = [h_1, h_2, \cdots, h_{\mathrm{T}}]$。预测值定义为 $\hat{z}_{1+\mathrm{T}}$。那么单步的时间序列预测模型可以定义为：

$$\hat{z}_{1+\mathrm{T}} = F(\boldsymbol{h}_{1:\mathrm{T}}, \theta) \tag{2-1}$$

其中 θ 是单步模型的参数，F 是单步预测模型的函数。

对于多步预测模型，我们假设需要预测 τ 步，定义 $\hat{\boldsymbol{z}}_{1+\mathrm{T}:1+\mathrm{T}+\tau} = [\hat{z}_{1+\mathrm{T}}, \cdots \hat{z}_{1+\mathrm{T}+\tau}]$。那么多步的时间序列预测模型可以定义为：

$$\hat{\boldsymbol{z}}_{1+\mathrm{T}:1+\mathrm{T}+\tau} = G(\boldsymbol{h}_{1:\mathrm{T}}, \Theta) \tag{2-2}$$

其中 Θ 是多步预测模型的参数，G 是多步预测模型的函数。

2.2 时间序列数据处理

2.2.1 时间序列数据划分

一般将原始的时间序列数据处理成适合监督学习的形式，主要有两种方法。一种是滑动窗口切分数据，假定我们有一个时间序列数据 $\boldsymbol{h} = [h_1, h_2, \cdots, h_{\mathrm{T}}]$，其中序列的长度为 T。我们假设我们需要的时间窗口大小为 τ，这样我们就可以切分出 $\mathrm{T} - \tau + 1$ 个训练数据，我们使用 $\boldsymbol{L}$ 来表示训练数据集合。下面我们使用数学公式简单描述下过程。

$$\begin{cases} L^1 = \boldsymbol{h}[1:\tau] \\ L^2 = \boldsymbol{h}[2:\tau+1] \\ \quad \cdots \\ L^{\mathrm{T}-\tau+1} = \boldsymbol{h}[\mathrm{T}-\tau+1:\mathrm{T}] \end{cases} \tag{2-3}$$

另外一种是不重叠的切分数据，同样我们假设有一个时间序列数据 $\boldsymbol{h} = [h_1, h_2, \cdots, h_{\mathrm{T}}]$，其中序列的长度是 T。我们假设我们训练的数据长度为 τ，这样我们可以切分出 $\frac{\mathrm{T}}{\tau}$ 个训练数据（假定我们能够完全切分），我们也用 $\boldsymbol{L}$ 来表示训练数据集合。下面我们使用数学公式简单描述下过程。

$$\begin{cases} L^1 = \boldsymbol{h}[1:\tau] \\ L^2 = \boldsymbol{h}[\tau+1:\tau+\tau] \\ \quad \cdots \\ L^{\frac{\mathrm{T}}{\tau}} = \boldsymbol{h}[\mathrm{T}-\tau+1:\mathrm{T}] \end{cases} \tag{2-4}$$

通常在实际中使用滑动窗口的方式来采样切分数据，这样不仅可以让模型接触连续的变化数据，同时也增大训练的数据集合大小，让神经网络模型更好地学习时间序列中的模式。

2.2.2 时序数据的预处理

原始采集的时间序列数据通常不是完整的，存在数据缺失、数据噪音以及数据不一致等问题。这类数据不能直接运用，因此我们需要对原始数据进行清理和预处理，时序预处理的方法有多种，包括缺失值处理、数据重采样、滑动平均处理等方法。

缺失值处理方法主要适用于原始时间序列中缺失部分数据，使用算法将这部分的数据填充上。实际工作主要使用的方法包括就近插补、线性插值、周期性 + 线性插值、统计数值填充等方法。就近插补，前推法，就是将当前时刻的观察值替换为缺失之前的最后一次观察值；后推法，使用缺失值后面的最早的观察值作为当前时刻的观察值。线性插值，如果我们的时间序列数据的变动有很强的趋势性，我们可以使用传统的数值插值方法，通过假定缺失部分是某个 N 次多项式曲线上的某些点，通过取缺失附近 N+1 个点数据，直接计算出该多项式曲线的参数，然后通过该多项式曲线估计出缺失位置的数值。周期性 + 线性插值，我们首先假设我们缺失部分是满足我们设定的周期，通过选择周期上其它的观察值，来计算该对应的 N 次多项式曲线，然后通过该多项式曲线来估计出缺失位置的数值。统计数值填充方法，根据缺失附近数据计算某个统计值将统计量用于该缺少时间点进行填充数值，比如使用缺失附近时间点的计算平均值，然后使用该平均值进行填充。

通常原始时间序列数据是一种时间间隔比较小采样生成的，存在许多的噪声同时我们需要使用其它时间间隔来预测结果。因此我们需要使用重采样方法，从原始的时间序列数据中按照预定时间间隔来进行采样统计处理。比如有一个按照分钟采样的原始时间序列数据集，我们需要按照小时时间间隔进行预测。因此我们需要做降采样操作，通过聚合一小时内的原始时间序列数据集合，并对该聚合体进行均值操作，用该均值代替当前小时时间点的时间序列数据。对于原始的时间序列采样间隔大于我们预测的间隔时，这时我们可以使用升采样方法，通过对原始间隔点间进行相应的线性插值处理，生成满足我们需要的时间序列数据。

在时间序列预测中，按每个时间点去观察趋势，可能有很大的噪音（震荡），所有我们往往需要按照一定时间间隔（叫做一个窗口），依照时间往前滑动处理。而移动平均方法通常可以通过平滑原始时间序列数据小的波动，突出时间序列中的长期趋势和周期性。通常有指数加权移动平均、简单移动平均、加权移动平均等移动平均方法。假设每个窗口有 k 个时间点，$\boldsymbol{w}$ 是每个窗口的观察值，

$\boldsymbol{h} = [h_1, h_2, \cdots, h_{\mathrm{T}}]$ 是时间序列的数据，那么简单移动平均可以表示为：

$$m_t = \frac{1}{k}\sum_{i=1}^{k} h_{t-i} \tag{2-5}$$

加权的移动平均可以表示为：

$$m_t = \frac{1}{k}\sum_{i=1}^{k} h_{t-i} \times w_i \tag{2-6}$$

对于指数加权移动平均方法，假定我们选择指数参数为 $\alpha, \alpha \in (0,1)$，那么 k 个时间点的指数加权平均移动可以表示为：

$$m_t = \frac{1}{k}\sum_{i=0}^{k-1} (1-\alpha)^i \times h_{t-i} \tag{2-7}$$

也可以近似为迭代方法：

$$m_t = \alpha x_t + (1-\alpha) m_{t-1} \tag{2-8}$$

通过调节 α 的大小，来调整当前值对窗口内远距离值的依赖程度，进而来控制平滑程度。

当元素数据预处理、切分后，我们还要需要做归一化处理。由于我们训练数据窗口是有限的，没有归一化的话，我们的网络必须每次都学习输入到输出尺度的缩放，然后对学习到的尺度缩放反向运用到输出中。这样神经网络不得不每次都学习缩放，不利于整个神经网络的收敛。同时神经网络期望输入值满足近似正态分布，通过归一化后也能提高神经网络训练的速度。常用的归一化有 Min-Max 归一化、Z-score 归一化、Box-Cox 归一化、Yeo-Johnson 归一化、自定义归一化等方法。时间序列数据表示为：$\boldsymbol{h} = [h_1, h_2, \cdots, h_{\mathrm{T}}]$，Min-Max 归一化是将特征的值映射到 $[0,1]$ 范围内，具体公式如下：

$$h'_t = \frac{h_t - \max(\boldsymbol{h})}{\max(\boldsymbol{h}) - \min(\boldsymbol{h})} \tag{2-9}$$

其中，$\max(\boldsymbol{h}) = \max([x_1, x_2, \cdots, x_{\mathrm{T}}])$，$\min(\boldsymbol{h}) = \min([x_1, x_2, \cdots, x_{\mathrm{T}}])$。Z-score 归一

化，将元素数据归一化均值为 0、方差为 1 的分布 $\sim N(0,1)$，具体公式如下：

$$\begin{cases} \overline{h} = \dfrac{1}{\mathrm{T}}\sum_{i=1}^{\mathrm{T}} x_i \\ \mathrm{var}(\boldsymbol{h}) = \dfrac{1}{\mathrm{T}-1}\sum_{i=1}^{\mathrm{T}}(h_i-\overline{h})^2 \\ h'_t = \dfrac{h_t-\overline{h}}{\mathrm{var}(\boldsymbol{h})} \end{cases} \tag{2-10}$$

Box-Cox [25] 归一化方法，对于一些长尾、偏度的数据分布，我们需要使用 Box-Cox 归一化方法将数据转换为近似正态分布和稳定的方差。Box-Cox 主要是针对正数值的数据样本，具体的数学公式如下：

$$h_t^{(\lambda)} = \begin{cases} \frac{h_t^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(h) & \text{if } \lambda = 0 \end{cases} \tag{2-11}$$

其中，参数 λ 可以通过对原始样本数据进行最大似然估计得出。

Yeo-Johnson [26] 归一化方法，同样也可以对长尾、偏度的数据进行转换，将数据转换为近似的正态分布和具有稳定的方差。当是 Yeo-Johnson 方法没有限制，可以对任意的数据样本进行转换，具体的数学公式如下：

$$h_t^{(\lambda)} = \begin{cases} [(h_i+1)^{\lambda}-1]/\lambda & \text{if } \lambda \neq 0, h_i \geq 0, \\ \ln(h_i+1) & \text{if } \lambda = 0, h_i \geq 0, \\ -[(-h_i+1)]^{2-\lambda}/(2-\lambda) & \text{if } \lambda \neq 2, h_i < 0, \\ -\ln(-h_i+1) & \text{if } \lambda = 2, h_i < 0 \end{cases} \tag{2-12}$$

同样 Yeo-Johnson 归一化的参数 λ 可以通过对原始样本数据进行最大似然估计得出。

2.3 深度学习相关理论

深度学习是机器学习的一个子集，它本质是一个三层或者多层的人工神经网络。这些神经网络通过模仿人类的大脑的行为，并允许从大量的数据中“学习”。虽然单层的神经网络仍然可以进行近似预测，但额外的隐藏层网络可以帮助优化和细化以提高模型的准确率。深度学习技术是日常生活和服务（如购物网站的商品推荐、智能家具语音终端、人脸打卡）以及新领域（如元宇宙、高级自动驾驶

技术）的基础。本小结主要阐述深度神经网络相关理论，具体包括前馈神经网络、卷积神经网络、循环神经网络、序列到序列神经网络等。

2.3.1 前馈神经网络

前馈神经网络由多层相互连接的节点组成，每一层都建立在前一层的基础上，以改进和优化数值回归或者分类。这种通过神经网络进行的计算称为前向传播。深度神经网络中一般都包含多层隐藏层，激活函数的非线性特征通过多个隐藏层的叠加，使得深度神经网能够对包含任意复杂非线性的数据进行建模并拟合。深度神经网络中输入和输出层被称作可见层，输入层是深度学习模型提取数据特征并进行处理的地方，输出层是最终预测或者分类的地方。

另外一个称为反向传播的过程使用梯度下降等算法[27,28]来计算损失误差，然后通过从后向前更新网络层的权重，来进行模型的学习。前向传播和后向传播一起允许神经网络做出预测并相应地纠正预测错误，随着时间的移动，算法能够越来越精准。

具体而言，假设我们有一个 3 个输入，1 个输出的神经元，其基本结构如图2-1。单个神经元的输入为 $\boldsymbol{h}=[h_1,h_2,\cdots,h_n]$，神经元的参数为 $\boldsymbol{m}=[m_1,m_2,\cdots,m_n]$，那

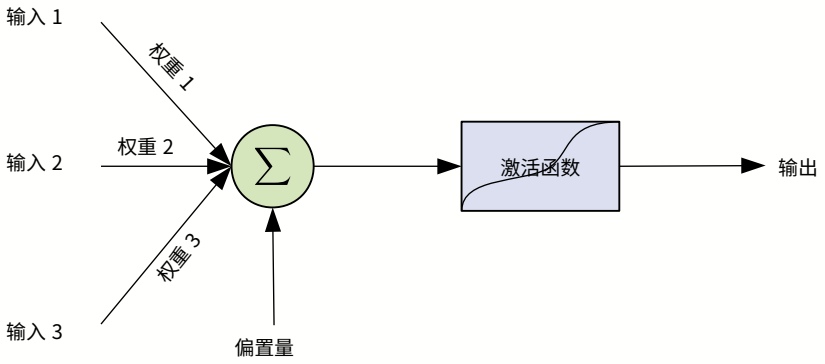


图 2-1 3 个输入，1 个输出的神经元示意图

么神经元的输出为：

$$y=f\left(\sum_i(h_iw_i)+b\right) \tag{2-13}$$

其中，函数 f 是一个非线性的激活函数（通常为 sigmod、tanh、relu 等函数），b 为神经元的偏置量。如果我们将不同的神经元一个接一个链接起来，一个神经元的输出作为后续一个神经元的输入，这样我们就形成了神经网络，如图2-2所示。假设我有 l 层的神经网络，定义第一层神经网络的激活函数为为 $f^{(1)}$，神经元的参数为 $\boldsymbol{w}^{(1)}$，偏置量为 $b^{(1)}$，输入为 $\boldsymbol{h}$，输出为 $\boldsymbol{z}^{(1)}$。第二层神经网络的激活函数为 $f^{(2)}$，

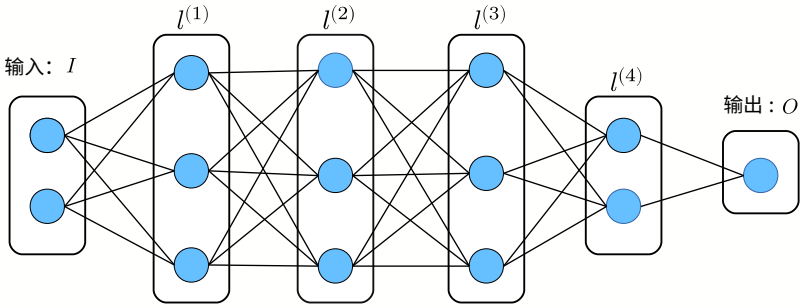


图 2-2 4 层神经元的神经网络模型示意图

偏置量为 $b^{(2)}$，神经元的参数为 $\boldsymbol{w}^{(2)}$，偏置量为 $b^{(2)}$。输入为 $\boldsymbol{z}^{(1)}$，输出为 $\boldsymbol{z}^{(2)}$。第 l 层神经网络的激活函数为 $f^{(l)}$，神经元的参数为 $\boldsymbol{w}^{(l)}$，偏置量为 $b^{(l)}$，输入为 $\boldsymbol{z}^{(l)}$，输出为 $\boldsymbol{z}^{(l+1)}$。其数学公式为：

$$\begin{cases} \boldsymbol{z}^{(1)} &= f^{(1)}\left(\boldsymbol{h}\cdot(\boldsymbol{w}^{(0)})^{\mathrm{T}}+b^{(0)}\right) \\ \boldsymbol{z}^{(2)} &= f^{(2)}\left(\boldsymbol{z}^{(1)}\cdot(\boldsymbol{w}^{(1)})^{\mathrm{T}}+b^{(1)}\right) \\ & \cdots \\ \boldsymbol{z}^{(l+1)} &= f^{(l+1)}\left(\boldsymbol{z}^{(l)}\cdot(\boldsymbol{w}^{(l)})^{\mathrm{T}}+b^{(l)}\right) \end{cases} \tag{2-14}$$

2.3.2 卷积神经网络

卷积神经网络是一种特殊的神经网络，它在图像处理、语音或者音频方面有卓越的性能。主要由卷积操作层、池化操作层和完全连接操作层等三个重要层组成，如图2-3所示。卷积层是卷积神经网络的第一层，虽然卷积层可以跟着额外的

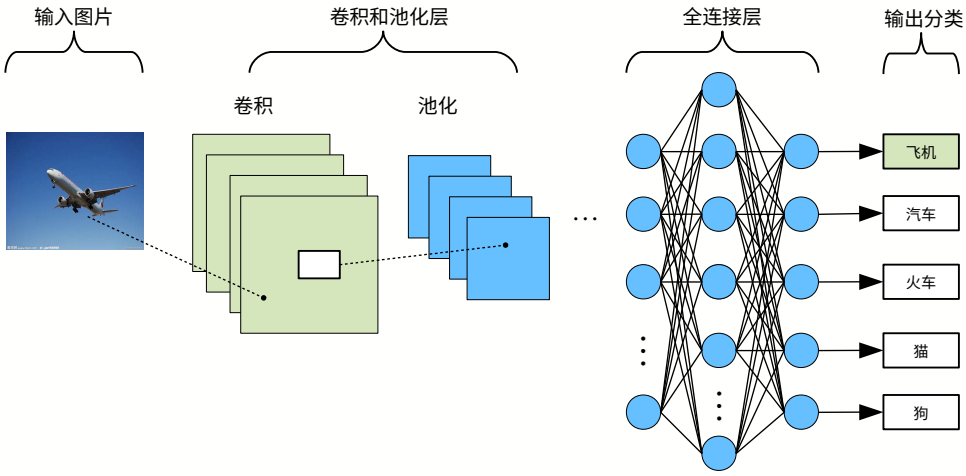


图 2-3 一个卷积神经网络示意图

卷积层或者池化层，但全连接层是网络的最后一层。随着网络的每一层，卷积神经网络 CNN 的复杂性不断增加，识别图像的更大部分。较早的层专注于简单的特征，例如颜色和边缘，随着图像的数据在卷积神经网络的各层不断前进，它开始识别物体的较大元素或形状，直到识别出预测的物体。

2. 普通卷积操作

卷积操作是卷积神经网络的最核心的运算，是大部分计算发生的地方。假设我们输入是一张 3 通道的图片，它由像素点矩阵组成，像素点主要由三色元组成，这意味着输入有三个维度（宽度、高度和通道数）。我们有一个卷积核函数，它在图像的中进行滑动卷积操作，检测图片的某个特征，这个过程被称为卷积。图像中的特征提取器基本为一个 2D 的权重数组。虽然卷积核函数的大小会发生变化，但是通常的大小为 3×3 矩阵，这也决定感受野的大小。然后将卷积核函数和图像的一部分像素进行点积，点积结果作为卷积操作的输出项。通过不断地在图像移动卷积核函数，生成最终的特征图，具体的卷积过程如图2-4所示。

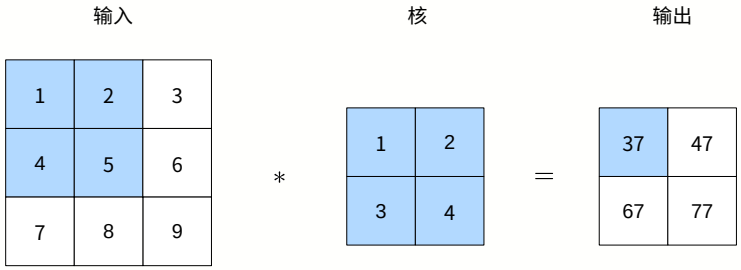


图 2-4 3×3 的图片在卷积核 2×2 操作下的输出过程示意图

通常在深度学习中，不使用原始的卷积操作，而是使用一个种叫做 cross-crrelation 的操作。下面我们通过数学公式来阐述该过程，假设输入的 RGB 图片为 $\boldsymbol{I}, \boldsymbol{I} \in \mathbb{R}^{m \times n \times 3}$，其中 m 是图片的高度，n 是图像的宽度。我们有一个核函数 $\boldsymbol{K}, \boldsymbol{K} \in \mathbb{R}^{p \times q \times 3}$，其中 p 是核的高度，q 是核的宽度。那么我们卷积运算结果可以用如下公式表示：

$$S(i,j) = (\boldsymbol{I} * \boldsymbol{K})(i,j) = \sum_{m} \sum_{n} \boldsymbol{I}(i+m)(j+n)\boldsymbol{K}(m)(n) \tag{2-15}$$

在图像中使用卷积操作主要有稀疏交互、参数共享和平移不变性等特征。通过使用比图像要小的多的卷积核函数，这以为这我们可以存储更少的参数，即能减少存储量和提高统计的有效性；同时也减少了计算量，加快模型的训练。

而卷积操作中每个图像输入的位置都被同一卷积核处理，使用共享参数的卷积核函数意味着我们不是为每个位置学习一组单独的参数，而只是学习一组参数。在参数共享的卷积操作特殊形式导致另外一个平移不变性的属性，如果有个一个

平移函数 $g(\cdot)$，一个卷积函数 $f(\cdot)$。有如下等式成立：

$$f(g(x)) = g(f(x)) \tag{2-16}$$

这意味着对原始输出进行 $f(\cdot)$ 操作后再进行 $g(\cdot)$ 操作，等价为先进行 $g(\cdot)$ 操作再 $f(\cdot)$。假设我们有一张灰度图 $\boldsymbol{I}$，$g(\cdot)$ 是个图片平移的函数，$\boldsymbol{I}'(x,y) = g(\boldsymbol{I}) = \boldsymbol{I}(x-1,y)$ 那么对平移后的图片 $\boldsymbol{I}'$ 做卷积操作，等效于对原始图像 $\boldsymbol{I}$ 进行卷积后再进行 $g(\cdot)$ 平移操作。当我们再处理时间序列数据时，卷积操作的平移不变性会产生一种时间轴的功能，如果我们将某个事件滞后，那么该事件的相同特征表示同样 i 也会滞后。对于处理的图像数据时，如果我们在输入图像中平移一部分，那么它的输出特征也会像输入图片一样平移。例如，在图像处理时，边缘检测是很有用的卷积神经网络的第一层。相同的边缘或多或少出现在图片中任何部分，因此在整个图像中共享这个边缘检测卷积核是现实的。

当卷积窗口形状为 1×1 的普通卷积操作，我们也称之为 1×1 卷积层，并将其中的卷积运算称为 1×1 卷积，如图2-5所示。使用了 1 大小的窗口，1×1 就不

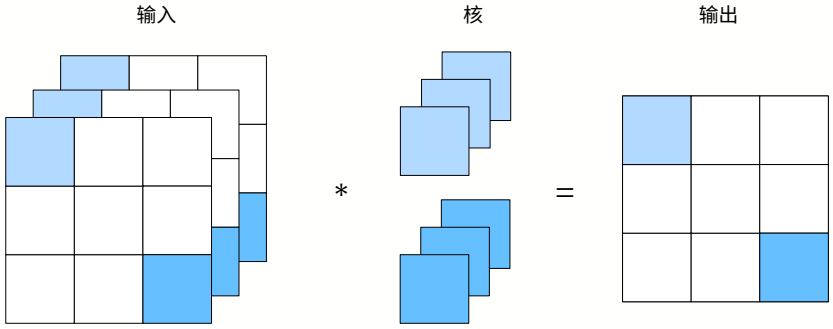


图 2-5 输入 3 通道的张量、输出 2 通道的张量的 1×1 卷积操作

能在应用到识别周围相邻元素构成的模式。实际上，1×1 卷积的计算主要发生在通道维数上，输出的中的每个元素来自输入中在高和宽上相同位置的元素在不同通道之间的按权重叠加。假如我们将通道维数当做特征维，将图片数据平铺成一维向量，那么 1×1 的卷积操作就可以等效于全连接层。通常在卷积网络中，1×1 卷积操作被当做保持长和宽不变的全连接层使用，起到通道维数降低或者上升的作用。

通常卷积的感受野是比较小的，只能注意它附近小范围的值。如果对输入图片卷积后，再使用池化，最后使用卷积，这样也可以加大卷积神经网络的感受野，但是池化操作会损失图片一部分信息。而空洞卷积[29]的提出，直接增大网络的感受野而不像池化操作那样损失信息，让每个卷积输出都包含较大范围的信息，如

图2-6。在图像处理中需要全局的信息捕获，那么可以运用空洞卷积进行处理。

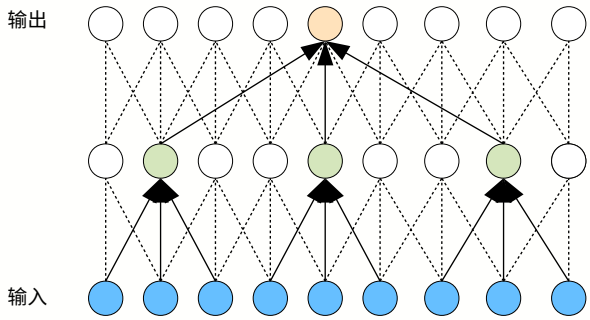
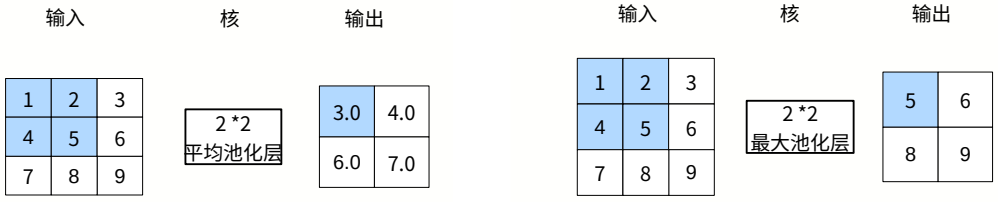


图 2-6 核大小为 3、空洞大小为 2 的空洞卷积操作示意图

3. 池化操作

池化层，也被称作下采样，进行空间维数降低，减少输入的参数量。池化操作将图像中某个位置的输出替换为它附近输出的统计输出，具体的池化过程如图2-7所示。



(a) 平均池化过程　　(b) 最大池化过程

图 2-7 池化操作示意图

和卷积操作一样，池化操作也是在输入中不断移动扫描，但不同之处在于这个操作没有任何权重。相反，池化核函数将聚合操作应用到对应池化的感受野中。池化操作主要有两种操作，最大池化、平均池化。最大池化操作，当池化操作在输入中移动扫描时，它总是在感受野中选择最大的像素值作为池化的结果。平均池化操作，同样在输入中移动扫描时，它将感受野中像素点的平均值作为池化的结果。通过池化操作后，有助于提高模型的视野，有助于降低模型的复杂性、提高计算效率并防止过拟合。

4. 全连接操作

全连接层，输出层中的每个节点都直接连接到前一层的节点，将前一层的输出作为输出层的输入。该层根据前几层提取的特征进行预测分类，全连接层通常使用 softmax 激活函数对输出信息，产生 0 到 1 的概率，用于预测分类。

2.3.3 循环神经网络

循环神经网络（RNN）是一种使用顺序数据作为输入的，一步步迭代训练的特殊神经网络，循环神经网络使用输入的顺序数据进行训练和预测。但是不像前馈神经网络或者卷积神经网络假设输入和输出都是相互独立的，循环神经网络的输出取决于序列中的先前元素，如图2-8所示。循环神经网络另外一个显著特征是

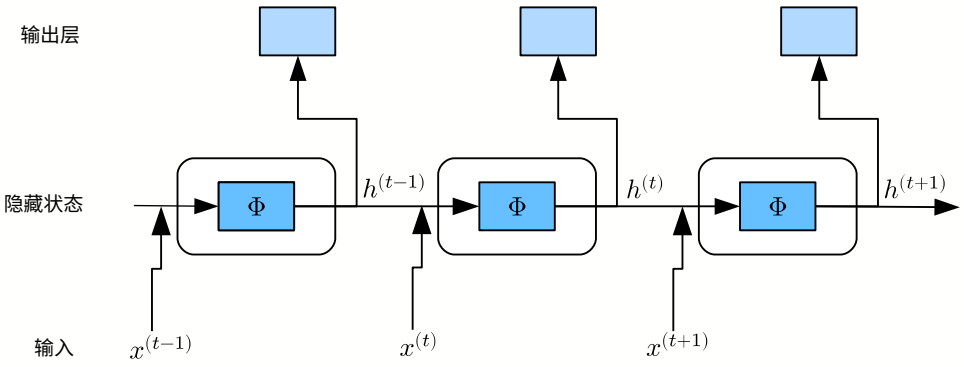


图 2-8 含隐藏状态的循环神经网络

它们在网络中的每一层都共享参数，虽然在前馈神经网络中每一层的网络的参数是不同的，但是循环神经网络在每一层中都共享同样的参数。定义一序列输入数据 $\boldsymbol{x}=[x^{(1)},x^{(2)},\cdots,x^{(n)}]$，其中 n 是序列的长度，定义循环神经网络的循环部分的参数为 $\boldsymbol{W}$，偏置为 b，输出的参数为 $\boldsymbol{V}$，偏置为 c，时刻 t 的隐藏状态为 $h^{(t)}$，初始隐藏状态为 $h^{(0)}$，输出为 $\boldsymbol{o}=[o^{(1)},o^{(2)},\cdots,o^{(n)}]$。则整个循环神经网络的运算过程如下述公式所示：

$$\begin{cases} h^{(1)}=\tanh(\boldsymbol{W}[h^{(0)},x^{(1)}]+b) \\ o^{(1)}=\mathrm{softmax}(\boldsymbol{V}h^{(1)}+c) \\ \quad\cdots \\ h^{(t)}=\tanh(\boldsymbol{W}[h^{(t-1)},x^{(t)}]+b) \\ o^{(t)}=\mathrm{softmax}(\boldsymbol{V}h^{(t)}+c) \end{cases} \tag{2-17}$$

循环神经网络使用时间反向传播算法（BPTT）算法来计算梯度和训练网络，它特定于到序列数据。时间反向传播算法的原理与经典的反向传播算法相同，模型计算从输出到输入的误差，然后用这个误差来训练神经网络模型。这些计算使我们能够适当地调整和拟合模型的参数，时间反向传播算法在每个时间步都对误差求和。

尽管通过 BPTT 算法来训练循环神经网络，但是它经常会遇到两个问题，即梯度消失和梯度爆炸。当梯度太小时，它会继续变小，更新权重参数直到它们变得微不足道，当这种情况发生时，模型将不再学习；当梯度太大时会发生梯度爆炸，模型的权重会变得很越来越大，模型出错无法训练。为了解决循环神经网络上述两个问题，Hochreiter S, Schmidhuber J. (1997) [30] 提出了长短记忆神经网络（简称为 LSTM），如图2-9所示。通过门控单元来控制传输状态，实现长距离的信息记忆。

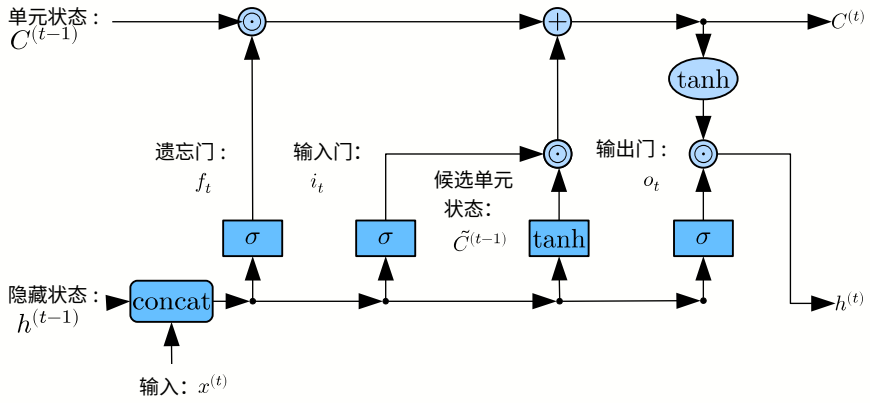


图 2-9 一个长短记忆网络示意图

原始的 RNN 循环神经网络只有一个隐藏状态 $h^{(t)}$，它对短期记忆有快速反应但对长期记忆反应迟钝。而 LSTM 神经网络通过增加一个状态 $C^{(t)}$，用它来保存长期记忆，我们通常称为单元状态（$C^{(t)}$）。LSTM 的核心就是这个单元状态，在网络中只有少数的线性运算作用到单元状态（$C^{(t)}$）上，所有单元状态（$C^{(t)}$）可以存储信息并且保存它们不怎么变化从而传递到很远的距离，这就是它能解决长期依赖的原因。LSTM 网络通常有三个门控单元，分别是遗忘门、选择门、输出门，通过门控操作来对单元状态（$C^{(t)}$）进行操作，如添加信息或删除信息。通过构建一个遗忘门，输入当前时刻的 $x^{(t)}$ 和上一时刻的输出 $h^{(t-1)}$，输出一个和 $C^{(t-1)}$ 同维度的向量 $f^{(t)}$，其中 $f^{(t)}$ 向量中 0 元素表示丢弃，1 元素表示完全保留。定义 $\boldsymbol{W}_f$ 是遗忘门的参数，b_f 是遗忘门的偏置项，$\sigma(\cdot)$ 是 sigmod 激活函数，用数学公式可以表示为：

$$f_t = \sigma(\boldsymbol{W}_f \cdot [h^{(t-1)}, x^{(t)}] + b_f) \tag{2-18}$$

选择门，用来决定更新哪些消息，其中 $\boldsymbol{W}_i$ 是选择门的参数，b_i 是选择门的偏置项，$\sigma(\cdot)$ 是 sigmod 激活函数，用数学公式可以表示为：

$$i_t = \sigma(\boldsymbol{W}_i \cdot [h^{(t-1)}, x^{(t)}] + b_i) \tag{2-19}$$

像传统的 RNN 神经网络一样，我们也需要生成一个新候选单元状态 $\widetilde{C}^{(t)}$，生成候选单元状态的网络参数为 $\boldsymbol{W}_c$，偏置项为 b_c，用数学公式可以表示为：

$$\widetilde{C}^{(t)} = \tanh(\boldsymbol{W}_c \cdot [h^{(t-1)}, x^{(t)}] + b_c) \tag{2-20}$$

通过对新的单元状态 $\widetilde{C}^{(t)}$ 和旧的单元状态 $C^{(t-1)}$ 进行更新，为当前的单元状态 $C^{(t-1)}$ 进行赋值。

$$C^{(t)} = f_t \odot C^{(t-1)} + i_t \odot \widetilde{C}^{(t)} \tag{2-21}$$

其中 $C^{(t-1)}$ 点乘 f_t 代表我们丢弃要遗忘的信息，$\widetilde{C}^{(t)}$ 点乘 i_t 代表我们从候选值向量中挑出要更新记忆的信息。最后，我们需要一个输出门，来决定输出什么结果。其中 $\boldsymbol{W}_o$ 是输出门的参数，b_o 是输出门的偏置项，$\sigma(\cdot)$ 是 sigmod 激活函数，用数学公式可以表示为：

$$o_t = \sigma(\boldsymbol{W}_o \cdot [h^{(t-1)}, x^{(t)}] + b_o) \tag{2-22}$$

通过将新的 $C^{(t)}$ 的输入函数 $\tanh(\cdot)$ 将所有的参数压缩到-1 到 1 之间的值。然后将其点乘输出门，我们就得到输出的部分。

$$h^{(t)} = o_t \odot \tanh(C^{(t)}) \tag{2-23}$$

一种由 Cho K, et al. (2014) [31] 提出的 LSTM 变种网络，名为 GRU 门控循环神经网络，如图2-10所示。它将 LSTM 长短记忆神经网络中的输入门控单元和遗忘门控单元控简化为一个更新门控单元，还将单元状态（cell state）和隐藏状态（hidden state）合并为一个状态，结构上比 LSTM 循环神经网络更简单，也是很流行的，门控循环单元中的遗忘门和更新门的输入均为当前时间步的输入 x_t 与上一时间步的隐藏状态 h_{t-1}，输出由激活函数 $\sigma(\cdot)$ 函数的全连接层计算得出。

具体来说，对于任意时间步的输入 $x^{(t)}$ 和上一步的隐藏状态 $h^{(t-1)}$。记忆门 z_t 和忘却门 r_t 的计算如下：

$$\begin{aligned} z_t &= \sigma(\boldsymbol{W}_z \cdot [h^{(t-1)}, x^{(t)}] + b_z) \\ r_t &= \sigma(\boldsymbol{W}_r \cdot [h^{(t-1)}, x^{(t)}] + b_r) \end{aligned} \tag{2-24}$$

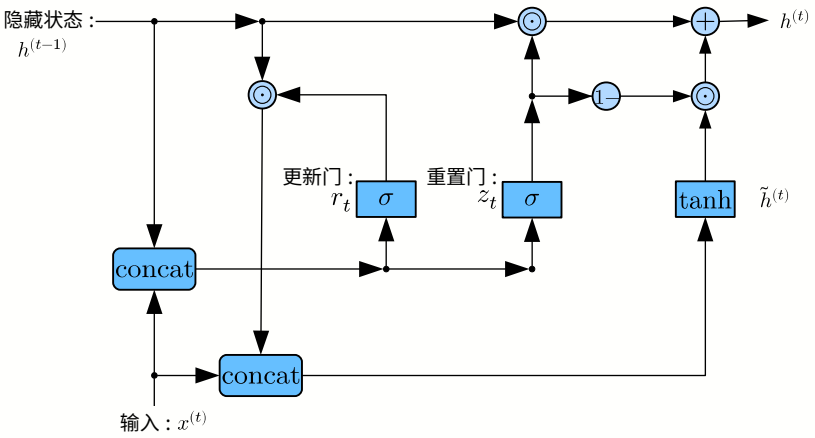


图 2-10 一个门控循环神经示意图

其中，$\boldsymbol{W}_z$ 和 $\boldsymbol{W}_r$ 是权重参数，$\sigma(\cdot)$ 函数可以将元素的值变化到 0 到 1 之间。因此，记忆门 z_t 和忘却门 r_t 中每个元素的值都是满足大于等于 0，小于等于 1 的。
接下来，门控循环单元将计算候选的隐藏状态，如公式2-25 所示：

$$\tilde{h}^{(t)} = \tanh(\boldsymbol{W}_h \cdot [r_t \odot h^{(t-1)}, x^{(t)}]) \tag{2-25}$$

我们将当前时间步忘却门的输出和上一个时间的隐藏状态 $h^{(t-1)}$ 做按元素乘法。如果忘却门中的元素值接近于 0，那么这意味着对应的忘却门状态元素为 0，即丢弃上一时间步的隐藏状态。如果忘却门的元素值接近于 1，那么这意味着对应的忘却门状态元素为 1，即保留上一时间步的隐藏状态。然后，将结果和当前时间步的输入进行连结，再通过含激活函数 tanh 的全连接层计算出候选的隐藏状态 $\tilde{h}^{(t)}$，其所有的元素值域是 $[-1, 1]$。
最后，间步 $h^{(t)}$ 的计算使用当前的时间步的记忆门来对上一时间步的隐藏状态 $h^{(t-1)}$ 和当前时间步的候选隐藏状态 $\tilde{h}^{(t)}$ 做组合：最后，上一时间步的隐藏状态 $h^{(t-1)}$ 和当前时间步产生的候选的隐藏状态 $\tilde{h}^{(t)}$ 分别与记忆们 z_t 进行点乘，最后输出线性组合。

$$h^{(t)} = z_t \odot h^{(t-1)} + (1 - z_t) \odot \tilde{h}^{(t)} \tag{2-26}$$

2.3.4 序列到序列神经网络

在机器学习任务，比如语言的翻译、多步时间预测等问题中，我们通常是输入一段序列数据，模型预测生成另外一段序列数据。为了实现端到端的神经网络模型训练，Sutskever I ,et al. (2014)[32] 提出 Seq2Seq 序列到序列的神经网络结构。Seq2Seq 是一种基于 LSTM 或者 GRU 的神经网络模型，主要包括两个部分，编码

器和解码器。如图2-11所示：

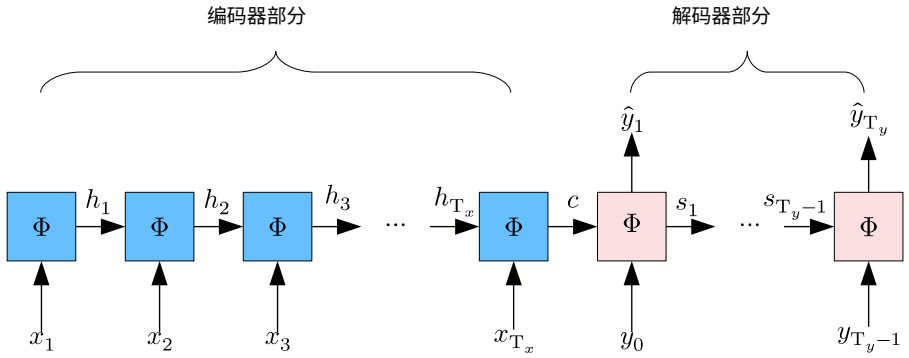


图 2-11 序列到序列网络示意图

1. 编码器

编码器通过每个时间步的输入序列，生成中间的背景向量 c。这个背景向量是保存这个输入序列的所有的相关信息，以便解码器使用这个相关信息进行信息解码。通常这个生成过程是通过循环神经网络来完成的，通过最后一个时间点的输出隐藏向量 h_{T_x} 作为背景向量。每个元素都是通过顺序输入到这个编码器，然后这个背景向量被网络更新，这个过程如图2-12所示。

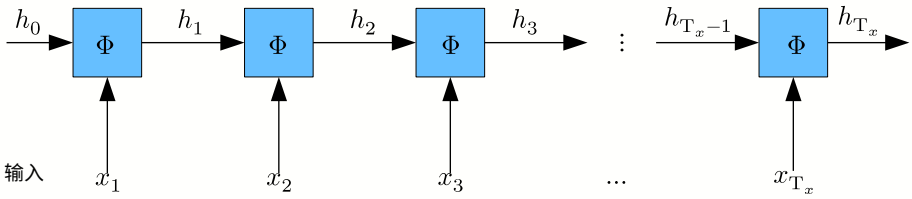


图 2-12 序列到序列神经网络编码器部分的示意图

2. 解码器

解码器在训练时，通过编码器生成的背景向量和对应的解码序列，对这个模型进行训练。通常解码器的第一时间点位置 y_{t_0} 是一个特殊的位置，用于告诉神经网络序列的开始，通过输入 y_{t_1} 时间点元素和背景向量 c 来预测时间点 t_2 的值 $\hat{y}_{t_2}$，通过损失函数 Q 来拟合真实的时间点 t_2 的值 y_{t_2}。通过这样不断地迭代，不断地训

练这个序列到序列模型。具体过程如公式2-29所示：

$$\begin{cases} (\hat{y}_{t_1}, \hat{s}_{t_1}) = D(y_{t_0}, c) \\ (\hat{y}_{t_2}, \hat{s}_{t_2}) = D(y_{t_1}, \hat{s}_{t_1}) \\ (\hat{y}_{t_3}, \hat{s}_{t_3}) = D(y_{t_2}, \hat{s}_{t_2}) \\ \cdots \\ (\hat{y}_{\mathrm{T}_y}, \hat{s}_{\mathrm{T}_y}) = D(y_{\mathrm{T}_y-1}, \hat{s}_{\mathrm{T}_y-1}) \end{cases} \tag{2-27}$$

具体的序列到序列的训练如图2-13所示，

最后通过如下公式2-28的损失函数进行训练：

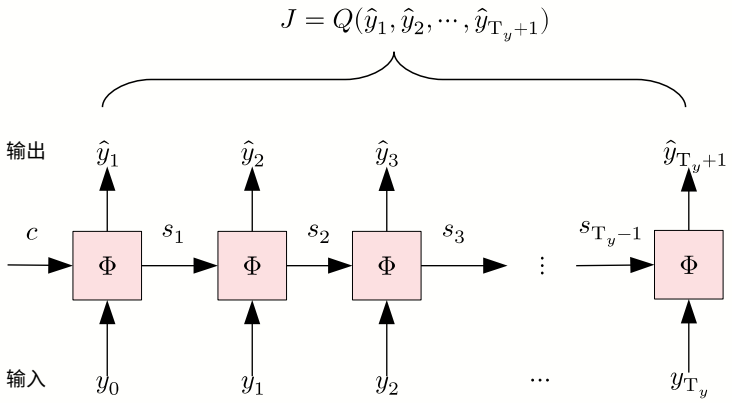


图 2-13 序列到序列模型中解码器预测示意图

$$J = Q(\hat{y}_{t_1,t_n}, y_{t_1,t_n}) \tag{2-28}$$

在推理的过程中，编码器是没有输入的，对于时刻 y_{t_1} 只能由编码器传入的背景向量 c 和默认的起始状态 y_{t_0}。通过 RNNCell 单元我们预测生成时刻 $\hat{y}_{t_1}$ 和状态 $\hat{s}_{t_1}$，然后通过迭代生成一段预测序列。如公式2-29所示，

$$\begin{aligned} (\hat{y}_{t_1}, \hat{s}_{t_1}) &= D(\hat{y}_{t_0}, c) \\ (\hat{y}_{t_2}, \hat{s}_{t_2}) &= D(\hat{y}_{t_1}, \hat{s}_{t_1}) \\ &\cdots \\ (\hat{y}_{\mathrm{T}_y}, \hat{s}_{\mathrm{T}_y}) &= D(\hat{y}_{\mathrm{T}_y-1}, \hat{s}_{\mathrm{T}_y-1}) \end{aligned} \tag{2-29}$$

具体的序列到序列的预测如图2-14所示：

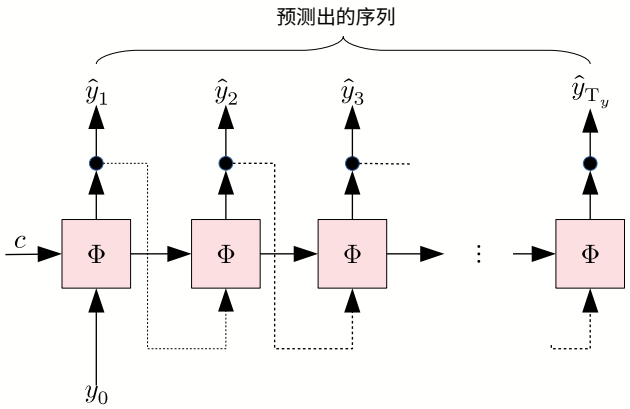


图 2-14 序列到序列模型中解码器预测示意图

3. 基于注意力的序列到序列

传统的 Seq2Seq 方法，背景向量是所有的输入序列通过编码器后生成的，而解码器是通过这个背景向量进行解码的。因此没法考虑这种情况，如果解码器中的输入 y_t 时刻点只依赖编码器中的某几个时间点的输入，而不是依赖所有的序列的时候。此，Bahdanau D, et al.(2014) [33] 提出了基于注意力的 Seq2SeqWithAttention 神经网络。不像传统的序列到序列的神经网络，编码器只生成一个背景向量 c，而是对每个位置都生成一个背景向量。这允许更多的信息存储在背景向量中，因为我们在编码器中的每个输入序列中都需要生成一个背景向量，但是我们只需要重要的信息也需要去掉重复的信息。也就是我们需要注意到重要的特征和表征。如图2-15所示，我们展示如何通过注意力机制来获得背景向量。

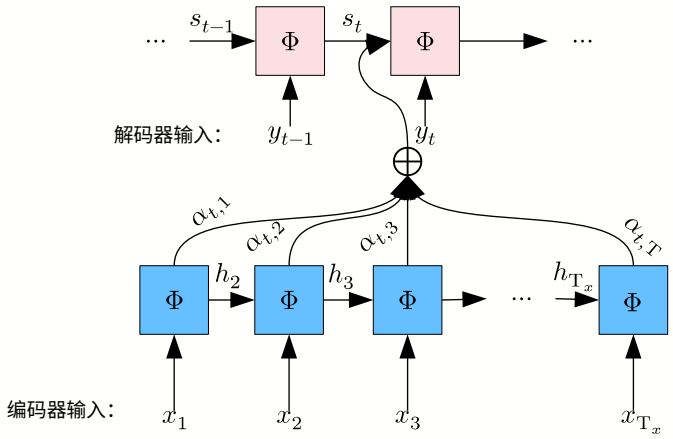


图 2-15 序列到序列模型中解码器预测示意图

背景向量 c_i 依赖于编码器中的一系列的编码器中编码输出的中间输出 $(h_1, h_2, \cdots, h_{T_x})$，每个中间输入 h_i 都包含对整个输入序列当前时间点 i 的信息。这个背景向量 c_i 是

通过对中间输出 $(h_1, h_2, \cdots, h_{T_x})$ 进行加权得出，如公式2-30计算出：

$$c_i = \sum_{j=1}^{\mathrm{T}_x} \alpha_{ij} h_j. \tag{2-30}$$

α_{ij} 是每个编码器的中间输出 h_j 加权系数，如公式2-31计算出：

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{\mathrm{T}_x} \exp(e_{ik})} \tag{2-31}$$

其中，

$$e_{ij} = \alpha(s_{i-1}, h_j)$$

α 是对齐的浅层神经网络，描述编码器中位置 j 和解码器中位置 i 的匹配程度。s_{i-1} 表示解码器中的 RNN 的隐藏输出状态。通过这一过程，模型就有注意力的机制。将解码器中元素和编码器中的元素对应起来，表达了解码器中的元素依赖了编码器中的某些序列片段的状态信息，而不是像传统的序列到序列模型只能依赖所有的序列状态信息。

2.4 本章小结

本章主要分为三部分，第一部分对时间序列数据的定义、时间序列预测问题的分类进行了阐述。第二部分主要阐述了时间序列数据处理的基本方法，包括数据的切分、预处理和归一化等问题。第三部分重点阐述了深度学习的相关理论基础，包括前馈神经网络、循环神经网络、序列到序列神经网络、卷积神经网络。

第三章 基于神经网络的多变量单步概率预测模型

针对多变量单步预测问题，本章提出了基于神经网络的多变量单步 GCNN-DeepAR 预测模型，该模型由多核门控卷积神经网络和 LSTM 长短记忆神经网络组成。模型首先对时间序列数据进行概率分布假设，然后利用概率分布损失函数训练模型学习时间序列概率分布中的参数集合，最后对该分布进行采样生成单步的预测值，并给出其预测的上界和下界。

3.1 多变量单步预测问题描述

在时间序列预测问题中，我们总是希望通过过去的历史时间序列数据和掌握的先验知识来预测未来时刻的数据。在传统的时间序列预测理论中，我们最简单的是预测未来某一时刻的数据，这被叫做单步预测问题。我们定义一段历史多变量时间序列数据 $\boldsymbol{h}=[\begin{bmatrix}h_1\\z_1\end{bmatrix},\cdots,\begin{bmatrix}h_{\mathrm{T}}\\z_{\mathrm{T}}\end{bmatrix}]$，其中 $h_t\in\mathbb{R}^{f_1}$，$z_t\in\mathbb{R}$。T 是历史时间序列数据的长度，f_1 是时间序列数据的特征变量数，z_t 是预测的目标值。同样我们也有一段对于的先验证知识 $\boldsymbol{x}=[x_1,\cdots,x_{\mathrm{T}},x_{\mathrm{T}+1}]$。我们希望预测未来 $\mathrm{T}+1$ 时刻的数据 $\hat{z}_{\mathrm{T}+1}\in\mathbb{R}$。这个单步预测的模型可以表示为：

$$\hat{z}_{\mathrm{T}+1}=G(\boldsymbol{h},\boldsymbol{x},\theta) \tag{3-1}$$

其中，θ 是这个单步预测模型的参数，模型的原型图如图3-1所示。

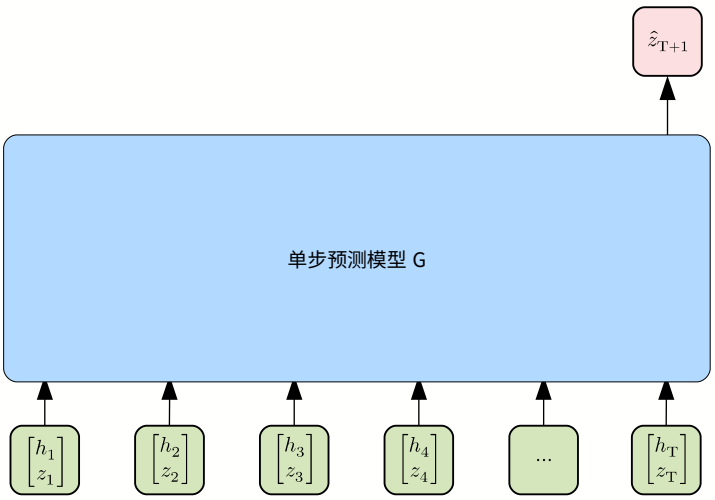


图 3-1 单步预测模型示意图

由于深度学习的不断发展，LSTM 长短记忆神经网络[30] 的提出，让循环神经网络可以运用到单步时间序列预测问题中，LSTM 长短记忆神经网络有效地避免了循环神经网络训练过程中梯度消失或爆炸问题，比较好的适用一定长度的序列输入。Flunkert V, et al.(2020)[34] 提出了 DeepAR 模型，它是一种针对时间序列的单步概率预测迭代模型，通过对历史的所有时序数据学习一个全局的模型。利用 LSTM 长短记忆神经网络来学习时间序列数据，并给出概率预测。通过 Monte Carlo 采样来预测时间序列数据；模型不假设高斯白噪声，可以根据数据的统计特征来假定概率分布。相对于传统的时间序列的点预测，通过区间预测来最小化决策的不确定性。

为了让深度学习运用到单步时间序列预测问题中来，我们需要构造我们的数据集，并且将问题转换为一个监督学习。我们使用滑动窗口的方法对原始的时间序列数据进行采样，其中采样的窗口大小为 $\mathrm{T}+1$。其中，输入的真实时间序列数据为 $\boldsymbol{h}=[\begin{bmatrix}h_1\\z_1\end{bmatrix},\begin{bmatrix}h_2\\z_2\end{bmatrix},\cdots,\begin{bmatrix}h_{\mathrm{T}}\\z_{\mathrm{T}}\end{bmatrix}]$；输出的真实时间序列的数据为 $\boldsymbol{z}=[z_2,z_3,\cdots,z_{\mathrm{T}+1}]$。通过输入序列和输出序列配对，让模型更好地学习单步预测迭代的过程；同时由于是滑动窗口采样，也能增大训练的数据集合大小，让神经网络更好地学习时间序列中的隐藏模式。

3.2 GCNN-DeepAR 概率预测模型

原始的 DeepAR[34] 模型，只是注意了全局的时间序列的变化，没有关注局部的时间序列的变化。针对模型缺失局部时序数据信息的捕获，因此我们先对时间序列数据进行一维门控多核卷积操作，捕获多尺度局部的时序信息依赖。对于模型假设的分布，其分布参数 $\boldsymbol{W}_{\mu}$ 和 $\boldsymbol{W}_{\sigma}$ 是在模型中是共享的，这样会导致模型分布概况的准确性。因此我们提出了全时域的分布参数，对于每个时间点都是一个参数，时间点间不共享分布参数，独立拥有分布参数，提高模型的表征能力。本章提出了 GCNN-DeepAR 模型，如图3-2所示。GCNN-DeepAR 模型分为三个部分，一维多核门控卷积层、LSTM 循环神经网络层、时间独立分布参数层。

3.2.1 输入预处理

由于我们模型的输入序列的长度有限，因此我们需要对我们的数据做转换操作。常用的转换有 z-scores 转换，将输入的数据归一化到均值为 0，方差为 1 的高

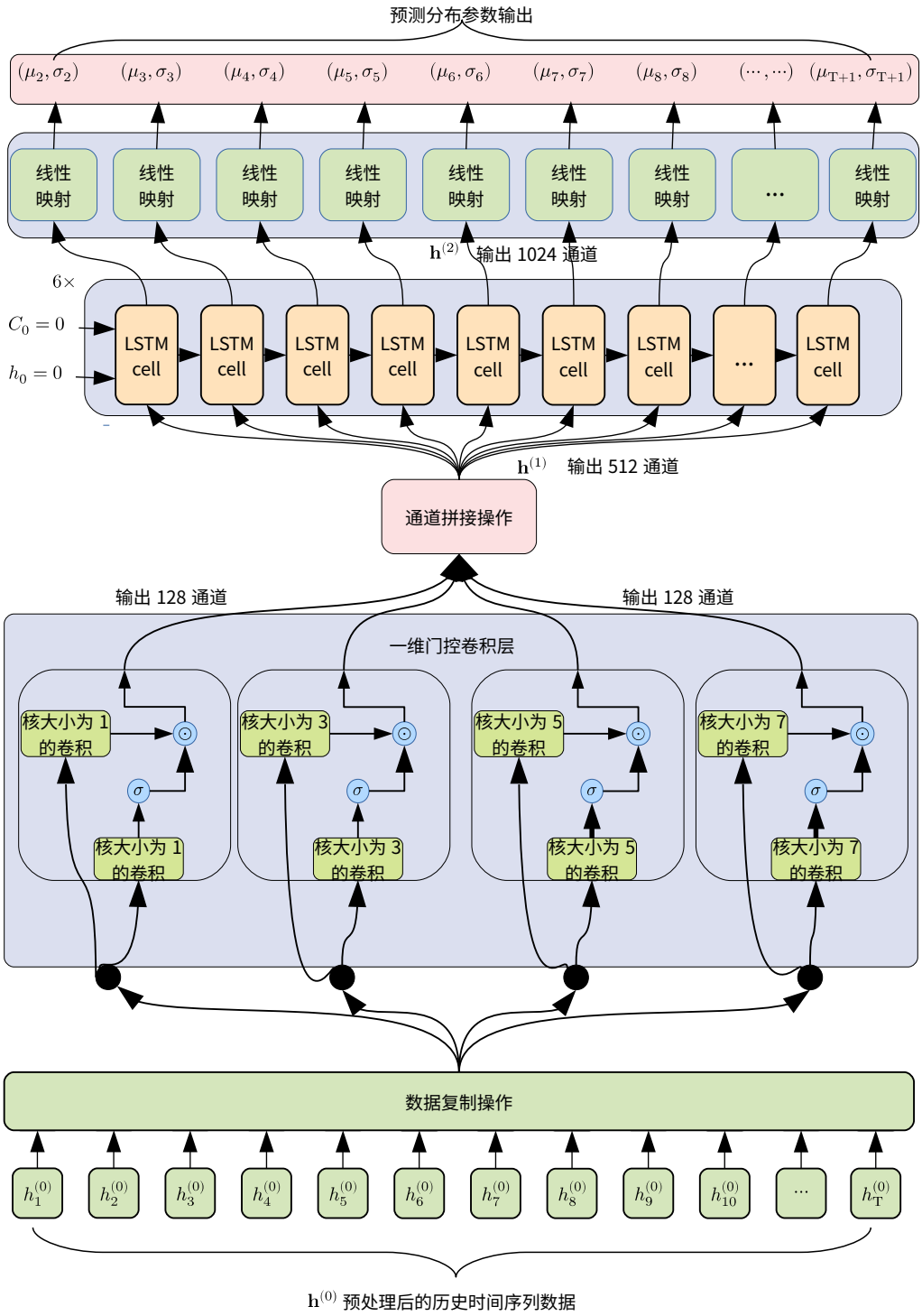


图 3-2 GCNN-DeepAR 模型的示意图，输入预处理后的历史时间序列数据 $\boldsymbol{h}^{(0)}$，输出高斯分布的参数集

斯分布，计算过程如公式3-2所示。

$$\begin{cases} \overline{h} = \dfrac{1}{\mathrm{T}}\sum_{t=1}^{\mathrm{T}} h_t \\ \mathrm{var}(\boldsymbol{h}) = \dfrac{1}{\mathrm{T}-1}\sum_{t=1}^{\mathrm{T}}(h_t-\overline{h})^2 \\ h_t^{(0)} = \dfrac{h_t-\overline{h}}{\mathrm{var}(\boldsymbol{h})} \end{cases} \tag{3-2}$$

min-max 转换，将特征的值映射到 [0,1] 范围内，计算过程如公式3-3所示。

$$h_t^{(0)} = \frac{h_t - \max(\boldsymbol{h})}{\max(\boldsymbol{h}) - \min(\boldsymbol{h})} \tag{3-3}$$

我们模型的输入序列的长度是有限的，然而部分时间序列的数据是有输入和输出尺度变化的。当输入和输出有尺度变化时，会造成我们的神经网络模型不容易训练，最终导致难以收敛。论文 DeepAR 提出了如下经验公式3-4。

$$\begin{cases} v_f = 1 + \dfrac{1}{\mathrm{T}}\sum_{t}^{\mathrm{T}} \boldsymbol{h}(t)_f \\ \boldsymbol{h}_f^{(0)} = \boldsymbol{h}_f / v_f \end{cases} \tag{3-4}$$

其中 $h(t)_f$ 是指输入历史时间序列时刻 t 的 f 特征变量。在实际运用中特征变量都是正的情况下，比较有效解决了尺度变化的问题。

通过选择合适的转换方法,将原始的时间序列数据 $\boldsymbol{h} = [\begin{bmatrix} h_1 \\ z_1 \end{bmatrix}, \begin{bmatrix} h_2 \\ z_2 \end{bmatrix}, \cdots, \begin{bmatrix} h_{\mathrm{T}} \\ z_{\mathrm{T}} \end{bmatrix}]$，转换为适合模型输入的时间序列数据 $\boldsymbol{h}^{(0)} = [h_1^{(0)}, h_2^{(0)}, \cdots, h_{\mathrm{T}}^{(0)}]$。

3.2.2 一维门控卷积

门控卷积 GatedCNN[35]，其核心在于为卷积的激活添加一个门控开关。通过门控单元，来决定时间序列数据中的短周期依赖信息多大概率传递到下一层的神经网络中。具体过程如图3-3所示。我们模型使用了 4 个门控卷积，其中每个门控卷积有两个卷积核，分别为 $(\boldsymbol{w}_1, \boldsymbol{v}_1)$，$(\boldsymbol{w}_2, \boldsymbol{v}_2)$，$(\boldsymbol{w}_3, \boldsymbol{v}_3)$，$(\boldsymbol{w}_4, \boldsymbol{v}_4)$。将输入序列 $\boldsymbol{h}^{(0)}$ 和一维卷积 $\boldsymbol{w}$ 进行操作得到 out1，将输入序列 $\boldsymbol{h}^{(0)}$ 和一维卷积 $\boldsymbol{v}$ 进行操作得到 out2。通过对 out2 做 $\sigma(\cdot)$ 操作，得到值属于 $[0,1]$ 的 gated。最后，将 gated 和 out1 进行点乘，让 $\boldsymbol{v}$ 卷积控制 $\boldsymbol{w}$ 卷积的输出。其中第一个门控卷积的卷积核 $(\boldsymbol{w}_1, \boldsymbol{v}_1)$ 的核大小是 1，输出的通道数是 128，如图3-4所示。第二个门控卷积的卷积核 $(\boldsymbol{w}_2, \boldsymbol{v}_2)$ 的核大小是 3，输出的通道数是 128，如图3-5所示。第三个门控卷积的卷积核 $(\boldsymbol{w}_3, \boldsymbol{v}_3)$ 的核大小是 5，输出的通道数是 128，如图3-6所示。第四个门控卷积的卷积核

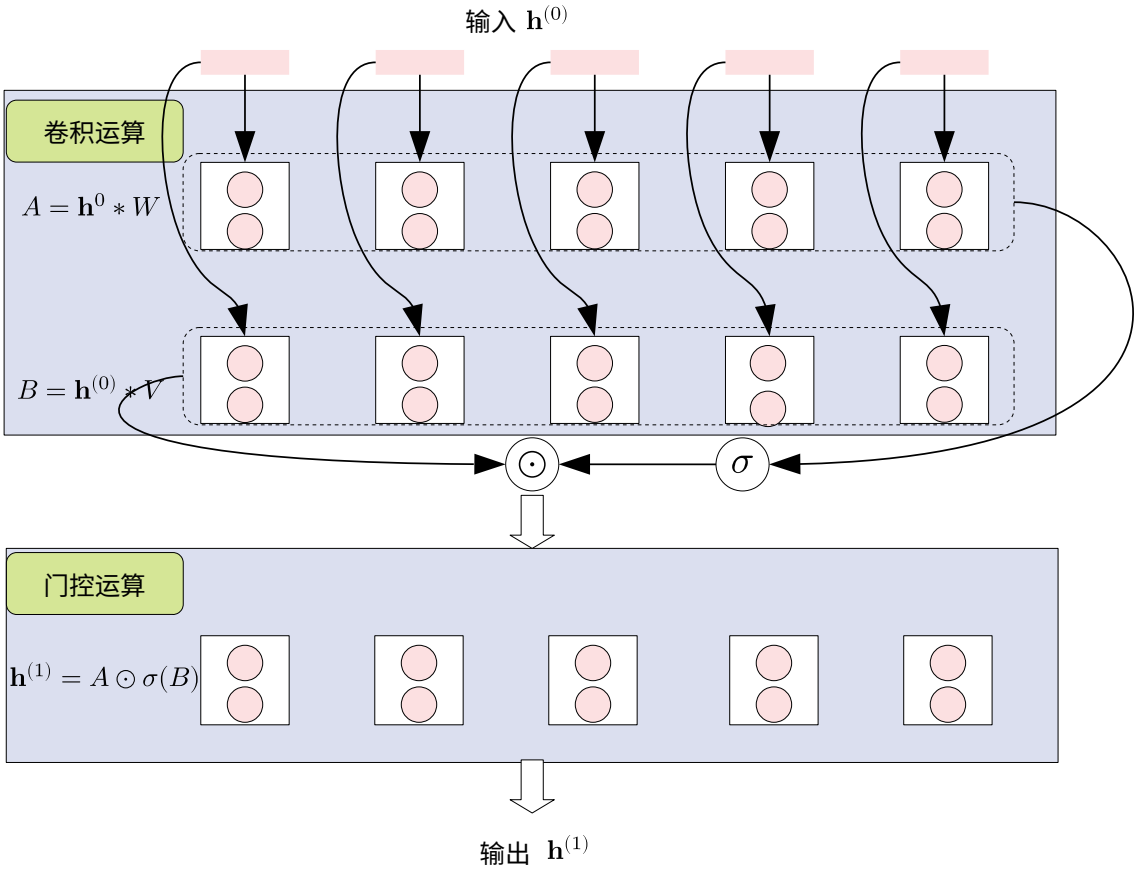


图 3-3 门控卷积操作示意图

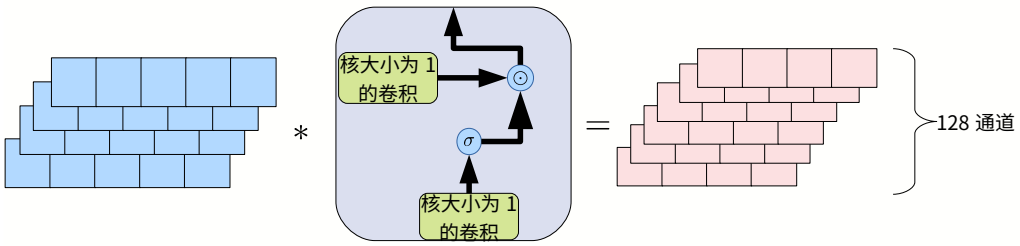


图 3-4 第一门控卷积示意图

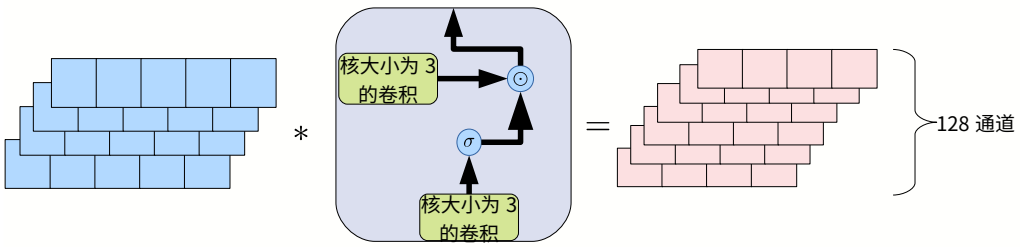


图 3-5 第二门控卷积示意图

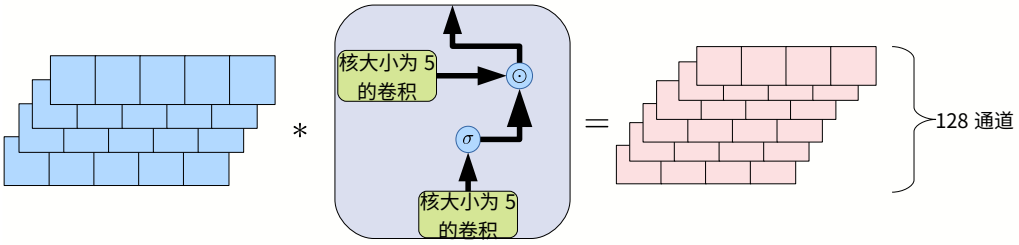


图 3-6 第三门控卷积示意图

($\boldsymbol{w}_4$，$\boldsymbol{v}_4$) 的核大小是 7，输出的通道数是 128，如图3-7所示。最后，将 4 个门控卷

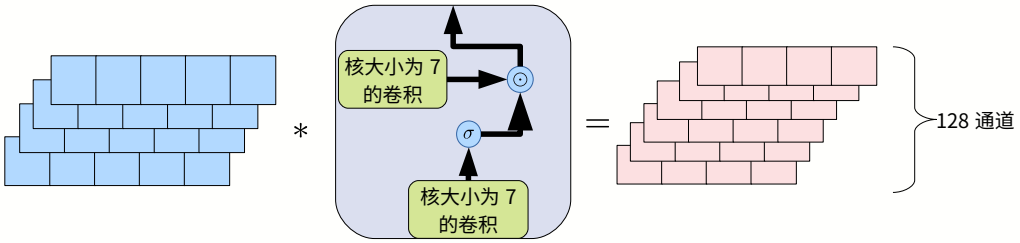


图 3-7 第四门控卷积示意图

积的输出拼接成 256 通道的时间序列数据 $\boldsymbol{h}^{(1)} = [h_1^{(1)}, \cdots, h_{\mathrm{T}}^{(1)}]$，其中 $h^{(1)} \in \mathbb{R}^{256}$，具体过程如公式3-5所示。

$$
\begin{aligned}
\mathrm{out}_1(i) &= (\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{w}_1(m)) \odot \sigma(\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{v}_1(m)) \\
\mathrm{out}_2(i) &= (\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{w}_2(m)) \odot \sigma(\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{v}_2(m)) \\
\mathrm{out}_3(i) &= (\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{w}_3(m)) \odot \sigma(\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{v}_3(m)) \\
\mathrm{out}_4(i) &= (\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{w}_4(m)) \odot \sigma(\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{v}_4(m)) \\
h^{(1)}(i) &= \mathrm{Concat}\left(\mathrm{out}_1(i), \mathrm{out}_2(i), \mathrm{out}_3(i), \mathrm{out}_4(i)\right)
\end{aligned}
\tag{3-5}
$$

其中，$\odot(\cdot)$ 表示矩阵间各元素相乘的运算，$\sigma(\cdot)$ 表示 sigmod 激活函数。

3.2.3 多层 LSTM 循环神经网络

我们使用 6 层叠加的 LSTM 循环神经网络，来捕获全局的时序依赖。其中初始单元状态输入 $C_0 = 0$ 和初始隐藏状态输入 $h_0 = 0$，隐藏状态的维数为 1024，通过 LSTM 循环神经网络学习序列中的全局依赖信息。输入一维门控卷积层的输出 $\boldsymbol{h}^{(1)}$，输出我们学习到的信息序列 $\boldsymbol{h}^{(2)} = [h_1^{(2)}, h_2^{(2)}, \cdots, h_{\mathrm{T}}^{(2)}]$，具体的过程如图 3-8所示。

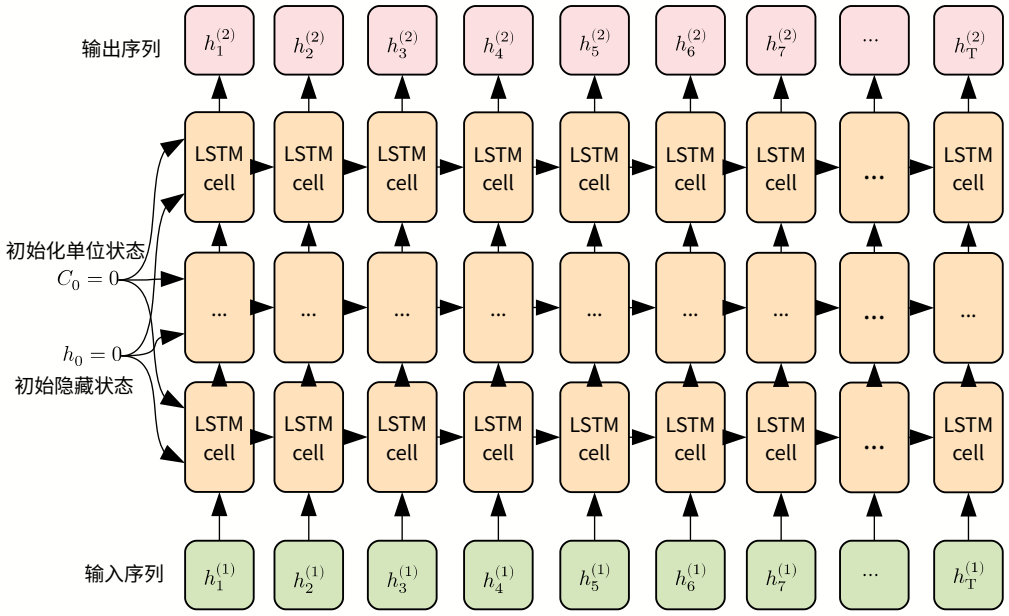


图 3-8 GCNN-DeepAR 模型的中循环神经模型示意图

3.2.4 时间独立分布参数层

我们假设时间序列是满足高斯分布的，当然我们也可以假设为其它参数分布（如负二项分布、指数分布等）。高斯分布通常由一对参数 (μ,σ) 来决定其分布。为了实现时间独立分布运算，我们有 T 组矩阵对 $\{(\mathrm{W}_{\mu}^{(i)} \in \mathbb{R}^{1024\times1}, \mathrm{V}_{\sigma}^{(i)} \in \mathbb{R}^{1024\times1})\}_{i=1}^{\mathrm{T}}$ 然后将这 T 组矩阵按照时间的顺序依次运用到 $\boldsymbol{h}^{(2)} = [h_1^{(2)}, \cdots, h_{\mathrm{T}}^{(2)}]$，生成 $\boldsymbol{\mu} = [\mu_2, \mu_3, \cdots, \mu_{\mathrm{T}+1}]$ 和 $\boldsymbol{\sigma} = [\sigma_2, \sigma_3, \cdots, \sigma_{\mathrm{T}+1}]$，如公式3-6计算所示。

$$
\begin{aligned}
\mu_2 &= \text{linear}(h_1^{(2)}, \mathrm{W}^{(1)}) * v_t \\
\sigma_2 &= \text{softplus}(\text{linear}(h_1^{(2)}, \mathrm{V}^{(1)})) * v_t \\
&\cdots \\
\mu_{\mathrm{T}+1} &= \text{linear}(h_{\mathrm{T}}^{(2)}, \mathrm{W}^{(\mathrm{T})}) * v_t \\
\sigma_{\mathrm{T}+1} &= \text{softplus}(\text{linear}(h_{\mathrm{T}}^{(2)}, \mathrm{V}^{(\mathrm{T})})) * v_t
\end{aligned}
\tag{3-6}
$$

其中，softplus 是个激活函数，让运算出来的 σ 满足大于零的约束。v_t 是目标值预处理后的尺度变换值。

3.2.5 训练和推理过程

对于概率预测问题，通常我们有个先验假设-时间序列分布是什么。通常对实际的问题，假设时间序列数据满足高斯分布。当然针对不同的场景，可能有更好的概率分布去描述时间序列的分布。在机器学习和深度学习中，通常是对损失函数进行极小化或者极大化进行优化训练参数的，传统的损失函数如 MSE 损失函数，如公式3-7。

$$J = \sum (z_i - \hat{z}_i)^2 \tag{3-7}$$

MAE 损失函数，如公式3-8。

$$J = \sum |z_i - \hat{z}_i| \tag{3-8}$$

MSE 和 MAE 损失函数，都是一种点估计。而分布损失函数是一种参数估计的方法，通过最大化似然函数，来训练估计其分布中的参数。我们定义高斯分布如公式3-9所示。

$$\ell_G(z|\mu,\sigma) = (2\pi\sigma^2)^{-1/2} \exp(-(z-\mu)^2/(2\sigma^2)) \tag{3-9}$$

其中，μ, σ 是我们需要估计的参数，并且需要 $\sigma > 0$ 的约束条件。

在训练过程中，我们有一段真实标注的时间序列数据 $\boldsymbol{z}_{2:\mathrm{T}+1} = [z_2, \cdots, z_{\mathrm{T}+1}]$，每个时间点 t 都可以得到一个概率 $\ell_G(z_t|\mu,\sigma)$。对于时间片段 $\tau = [2, 3, \cdots, \mathrm{T}+1]$，我们可以得到其联合似然函数 J，如公式3-10所示。

$$J = -\sum_{t=2}^{\mathrm{T}+1} \log\left(\ell_G(z_t|\mu_t,\sigma_t)\right) \tag{3-10}$$

通过反向传播算法来极小化这个损失 J，实现单步概率模型的学习，如图3-9所示。

在推理时，GCNN-DeepAR 单步预测模型的输出 ($\boldsymbol{\mu}$,$\boldsymbol{\sigma}$)，取出最后一个位置的参数组 ($\mu_{\mathrm{T}+1}$,$\sigma_{\mathrm{T}+1}$)。然后对高斯分布 $(\mu_{\mathrm{T}+1}, \sigma_{\mathrm{T}+1})$ 进行 Monte Carlo 采样，采样出 $\hat{z}_1$ 作为模型 $\mathrm{T}+1$ 时刻的预测值，如图3-10所示。

3.3 实验过程与结果

3.3.1 实验内容介绍

实验软硬环境：本章实验的硬件环境为包括 Intel Xeon E5 处理器，NVIDIA GeForce RTX 3090 24G 显存，256G 内存的台式服务器。本章的所使用的软件环境为 NVIDIA-cuda 11.4，python 3.7.11，深度学习框架 pytorch-GPU 1.10.0，机器学习

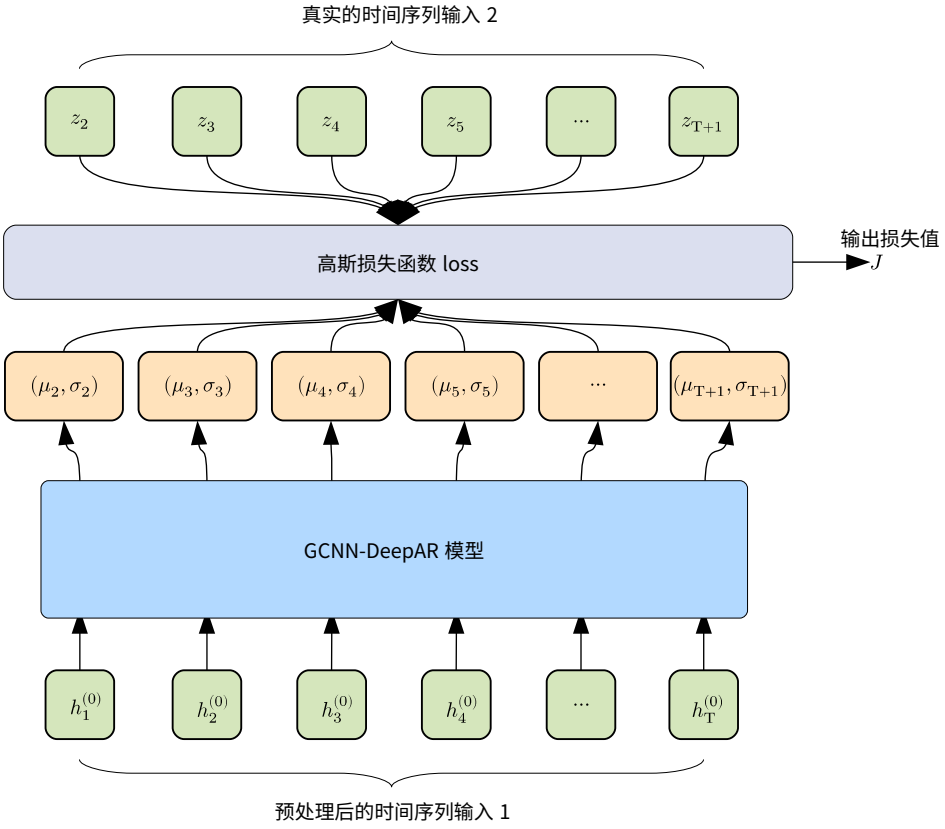


图 3-9 GCNN-DeepAR 模型的训练过程示意图

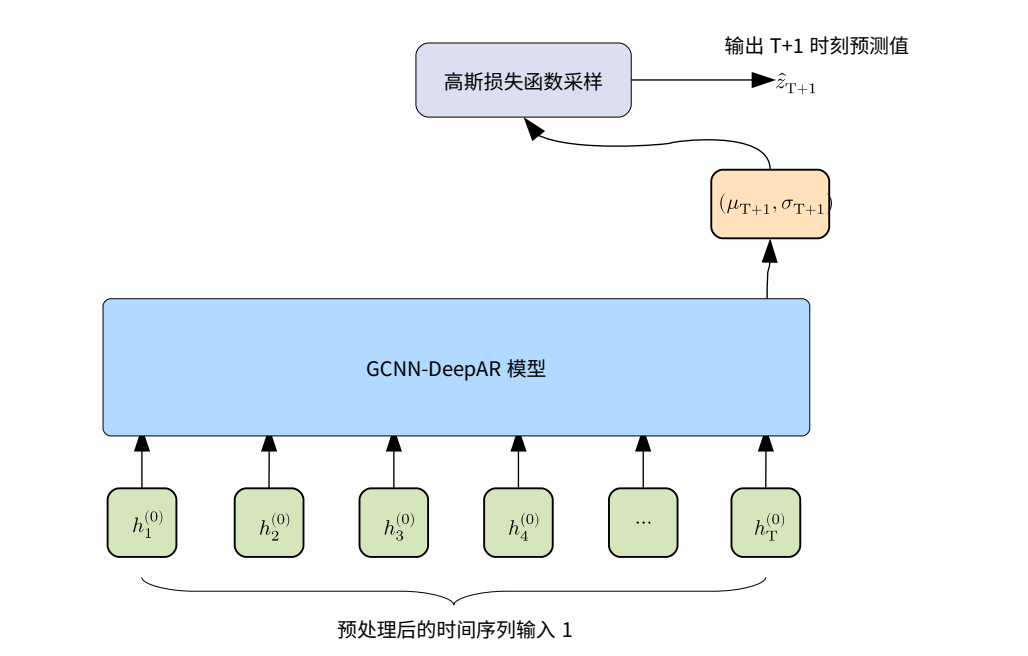


图 3-10 GCNN-DeepAR 模型的推理过程示意图

框架 sklearn 1.0.2，数据处理框架 pandas 1.3.5。操作系统为 18.04.6 LTS，内核版本为 5.11.0-41-generic x86_64。

训练参数：本文中 GCNN-DeepAR 模型的需要参数，如表3-1。

表 3-1 参数设置

模型参数	数值形式
训练批次 (batch size)	64
训练次数 (epoch)	150
数据窗口大小 (window size)	100
初始化学习率	0.001
优化方法	adam
GatedCNN 卷积核	[1, 3, 5, 7]
GatedCNN 卷积输出维数	[64, 64, 64, 64]
GatedCNN 卷积核边界填充	0
LSTM 层数	6
LSTM 隐藏层维数输入维数	512
LSTM 隐藏层维数输出维数	1024
时间独立分布参数层输入维数	1024
时间独立分布参数层输出维数	1
训练集比例	0.8
测试集比例	0.1
验证集比例	0.1

实验数据：本实验主要使用两个数据集，空气污染 PM2.5 时间序列数据集和某公司一段时间的股票时间序列数据集。

空气污染 PM2.5 时间序列数据集[36]，是收集 2020-01-02:00 到 2020-01-22:23 每个小时的污染 PM2.5 数据，总共 504 条数据。我们使用 479 条数据作为训练集，最后 25 条数据作为测试集。空气污染 PM2.5 数据集主要包括 PM2.5 值、露点、温度、气压、风速、风向六个特征变量，我们利用历史数据进行 PM2.5 值的预测。图3-11是该数据集的时序图。某上市公司的股票时间序列数据集，是收集从 2019-09-09 到 2022-03-01 每天的股票数据，总共 598 条数据。我们使用 572 条数据作为训练集，最后 26 条数据作为测试集。股票数据集主要包含开盘价、最高价、最低价、收盘价、交易量五个特征变量，我们利用历史数据进行收盘价的预测。图3-12是该数据集的时序图。

模型评估指标：本章使用常见的评估指标 MSE，RMSE，ND，ρ-risk 指标。假如真实的时间序列为 $z_{i,t}$，模型预测的中位数为 $\hat{z}_{i,t}$。那么时间序列评估指标 MSE

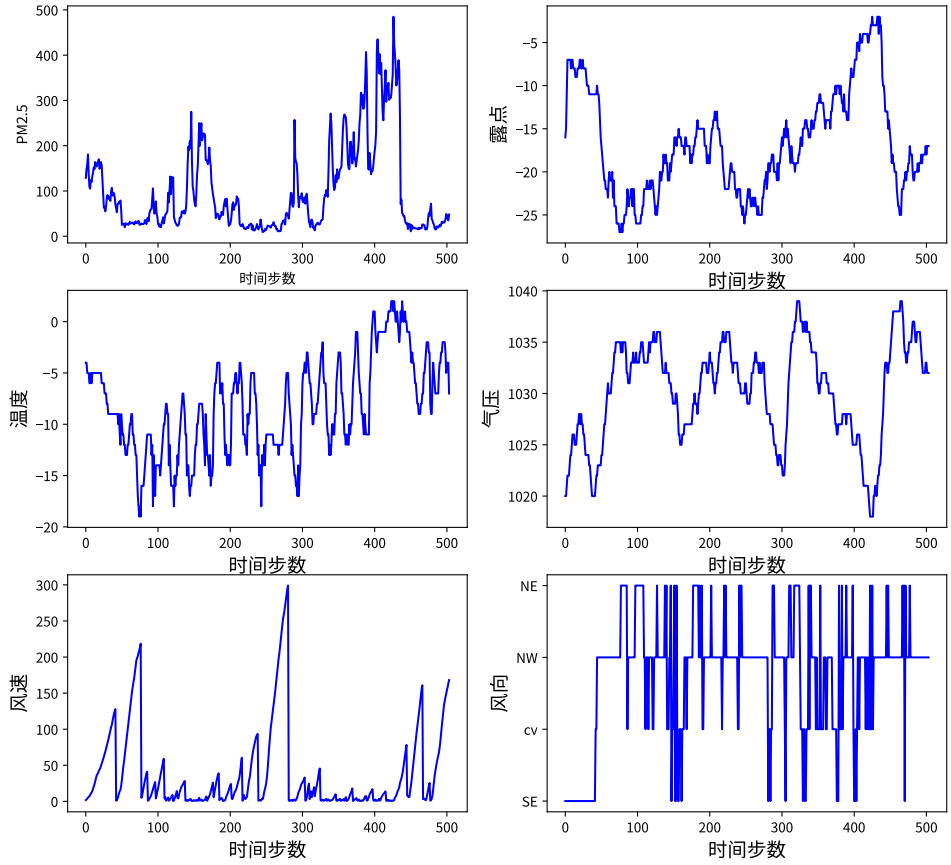


图 3-11 空气污染 PM2.5 从 2020-01-02:00 到 2020-01-22:23 的时序图

如公式3-11计算：

$$\mathrm{MSE}=\frac{\sum_{i,t}\left(z_{i,t}-\hat{z}_{i,t}\right)^{2}}{\sum_{i,t}\left|z_{i,t}\right|} \tag{3-11}$$

指标 RMSE 如公式3-12计算：

$$\mathrm{RMSE}=\frac{\sqrt{\frac{1}{\mathrm{T}}\sum_{i,t}\left(z_{i,t}-\hat{z}_{i,t}\right)^{2}}}{\frac{1}{\mathrm{T}}\left|\sum_{i,t}z_{i,t}\right|} \tag{3-12}$$

指标 ND 如公式3-13计算：

$$\mathrm{ND}=\frac{\sum_{i,t}\left|z_{i,t}-\hat{z}_{i,t}\right|}{\sum\left|z_{i,t}\right|} \tag{3-13}$$

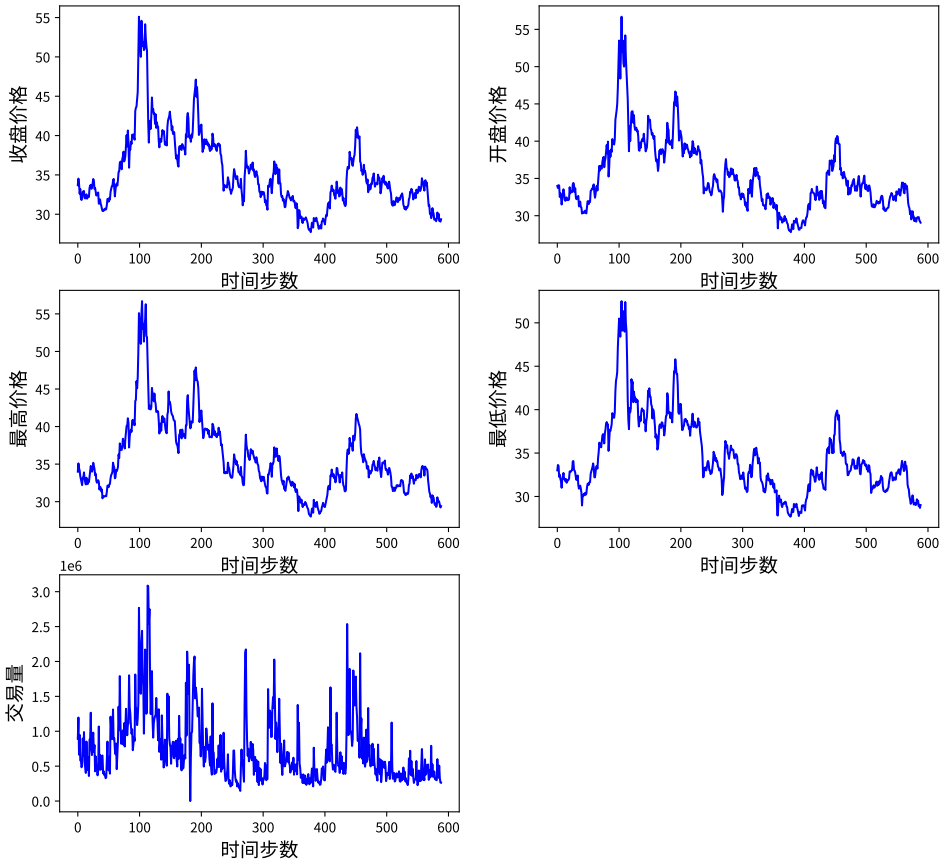


图 3-12 某上市公式从 2019-09-09 到 2022-03-01 的股票时序图

我们指定一个分数 ρ，模型对应预测的序列为 $\hat{z}_{i,t}^{\rho}$，指标 ρ-risk 如公式3-14计算：

$$\rho\text{-risk} = \frac{2 * \sum_{i,t}\left\{(\hat{z}_{i,t}^{\rho} - z_{i,t})_{+} \cdot (\rho - 1) + (z_{i,t} - \hat{z}_{i,t}^{\rho})_{+} \cdot \rho\right\}}{\sum_{i,t}|z_{i,t}|} \tag{3-14}$$

3.3.2 实验数据集分析与处理

对于空气污染 PM2.5 数据集，包括 PM2.5、露点、温度、气压、风速、风向这六个参数，其中风向是类别性，其他都是数值性。其中 PM2.5 是我们时间序列需要预测的目标值。

对于某上市公司股票数据集，包括开盘价、收盘价、最高价、最低价、交易量这五个参数，这五个参数都是数值性，其中收盘价是我们时间序列需要预测的目标值。

1. **数据切分**

对于空气污染 PM2.5 数据集，训练时我们使用滑动窗口方法对时间序列数据集进行数据采样，默认我们使用窗口大小为 T = 25。其中使用数据切片 $\boldsymbol{h}[1:24]$ 作为模型的输入序列，数据 $\boldsymbol{z}[2:25]$ 作为时间序列预测的真实序列（只包括 PM2.5 值）。对于预测时，我们同样使用滑动窗口进行数据采样，此时我们的窗口大小为 T = 25，通过模型预测生成未来时刻 $t = 1$ 的预测目标值。

对于某上市公司的股票时间序列数据集，训练时我们使用滑动窗口方法对真实的数据进行采样，默认我们使用的窗口大小为 T = 13。其中使用数据切片 $\boldsymbol{h}[1:12]$ 作为模型的输入时间序列，数据切片 $\boldsymbol{z}[2:13]$ 作为时间序列预测的真实序列（只包含收盘价）。对于预测时，我们也使用滑动窗口方法进行采样，此时我们的窗口大小设定为 T = 13，通过模型生成未来时刻 $t = 1$ 的预测值。

2. **数据预处理**

在真实时间序列数据中，时间序列有一定的趋势性。我们的神经网络模型的输入序列的长度是有限，那么在每个输入窗口中我们的神经网络模型都需要学习输入值和输出值的尺度变化，这样会导致我们的神经网络损失函数很难收敛，最终模型无法训练。因此我们需要一种尺度转换和反转换的方法，本实验直接使用 deepAR [34] 论文的经验尺度转换公式3-15和经验反转换公式3-16。实验表明经验尺度转换公式3-15具有很好的作用，加快神经网络的收敛。

$$\begin{cases} v_f = 1 + \dfrac{1}{\mathrm{T}} \sum\limits_{t=1}^{\mathrm{T}} \boldsymbol{h}(t)_f \\ \tilde{\boldsymbol{h}}_f = \boldsymbol{h}_f / v_f \end{cases} \tag{3-15}$$

$$\begin{cases} \mu_t = v_t \left(\boldsymbol{W}_t^{\mathrm{T}} \boldsymbol{h}_t + b_t \right) \\ \sigma_t = (1 + \exp(\boldsymbol{V}_t^{\mathrm{T}} \boldsymbol{h}_t + b_t)) v_t \end{cases} \tag{3-16}$$

3.3.3 对比实验组

本文中我们使用五个模型进行对比：

1. 普通的 LSTM 长短记忆神经网络模型：使用 MSE 损失函数，使用相应的尺度变化方法对原始的时间序列数据进行预处理。

2. LSTNet [37] 长期和短期时间序列网络模型：使用核大小为 3 的普通的卷积神经网络捕获短周期，LSTM 长短记忆神经网络捕获长周期。使用 MSE 损失函数，使用相应的尺度变化方法对原始的时间序列数据进行预处理。

3. TCN[38] 时间卷积神经网络模型：使用多个膨胀卷积叠加形成神经网络，其中卷积的核大小为 3，空气污染 PM2.5 预测中的膨胀卷积集合的膨胀列表为 $[1,2,4,8]$、股票预测中膨胀卷积集合的膨胀列表为 $[1,2,4]$。使用 MSE 损失函数，使用相应的尺度变化方法对原始的时间序列数据进行预处理。

4. 原始的 DeepAR[34] 自回归模型：使用高斯分布损失函数，使用相应的尺度变化方法对原始的时间序列数据进行预处理。DeepAR 是一种概率预测模型，我们可以生成 $[\mu-\sigma,\mu+\sigma]$、$[\mu-2\sigma,\mu+2\sigma]$、$[\mu-3\sigma,\mu+3\sigma]$ 区间的预测数据。能更好地捕获全局模式，区间预测能为预测的不确定性提供更好的量化保证。

5. GCNN-DeepAR 自回归模型：使用高斯分布损失函数，使用相应的尺度变化方法对原始的时间序列数据进行预处理。通过模型改进，让模型更好地捕获短周期的依赖和复杂时间序列分布的适应性。

3.3.4 预测效果分析

在空气污染 PM2.5 数据集上，我们进行单步预测。从表3-2可知，当单步预

表 3-2 空气污染 PM2.5 数据集对比实验结果

模型	预测步长 τ	指标					
		MSE	RMSE	ND	ρ-risk-10	ρ-risk-50	ρ-risk-90
LSTM	$\tau=1$	3.25259	0.06611	0.23953	-	0.26120	-
LSTNet	$\tau=1$	4.15120	0.07469	0.29584	-	0.32596	-
TCN	$\tau=1$	2.30067	0.05560	0.16718	-	0.18112	-
DeepAR	$\tau=1$	5.91764	0.08918	0.29670	0.17499	0.31211	0.12460
GCNN-DeepAR	$\tau=1$	2.01320	0.05202	0.19448	0.19690	0.21859	0.11902

测时，GCNN-DeepAR 模型比原始的 DeepAR 模型在指标 RMSE、ND、ρ-risk-10、ρ-risk-50、ρ-risk-90 都有所提高。为了更为直观地描绘 GCNN-DeepAR 模型和原始的 DeepAR 模型的预测，图3-13表示 GCNN-DeepAR 模型在预测步长 $\tau=1$ 的结果，其中浅绿色的上下界为 $[\mu-2*\sigma,\mu+2*\sigma]$ 表出，μ 和 σ 是模型预测出下一步概率分布的参数。图3-14表示原始的 DeepAR 模型在预测步长 $\tau=1$ 的结果。根据表3-2和图3-13-图3-14可知，GCNN-DeepAR 模型在单步预测时，RMSE，ND，ρ-risk-10，ρ-risk-50，ρ-risk-90 等指标都有提高，证明了 GCNN-DeepAR 模型预测精度有很好的提高。

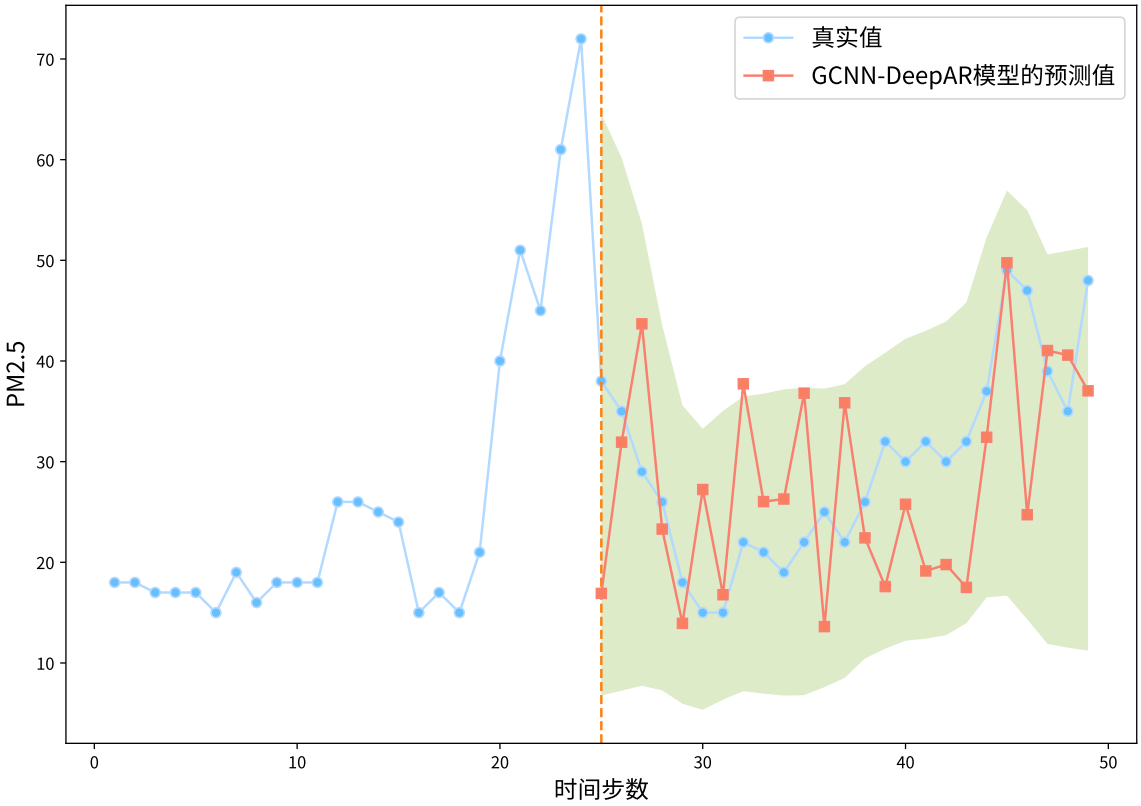


图 3-13 GCNN-DeepAR 模型在预测步长 $\tau = 1$ 的结果

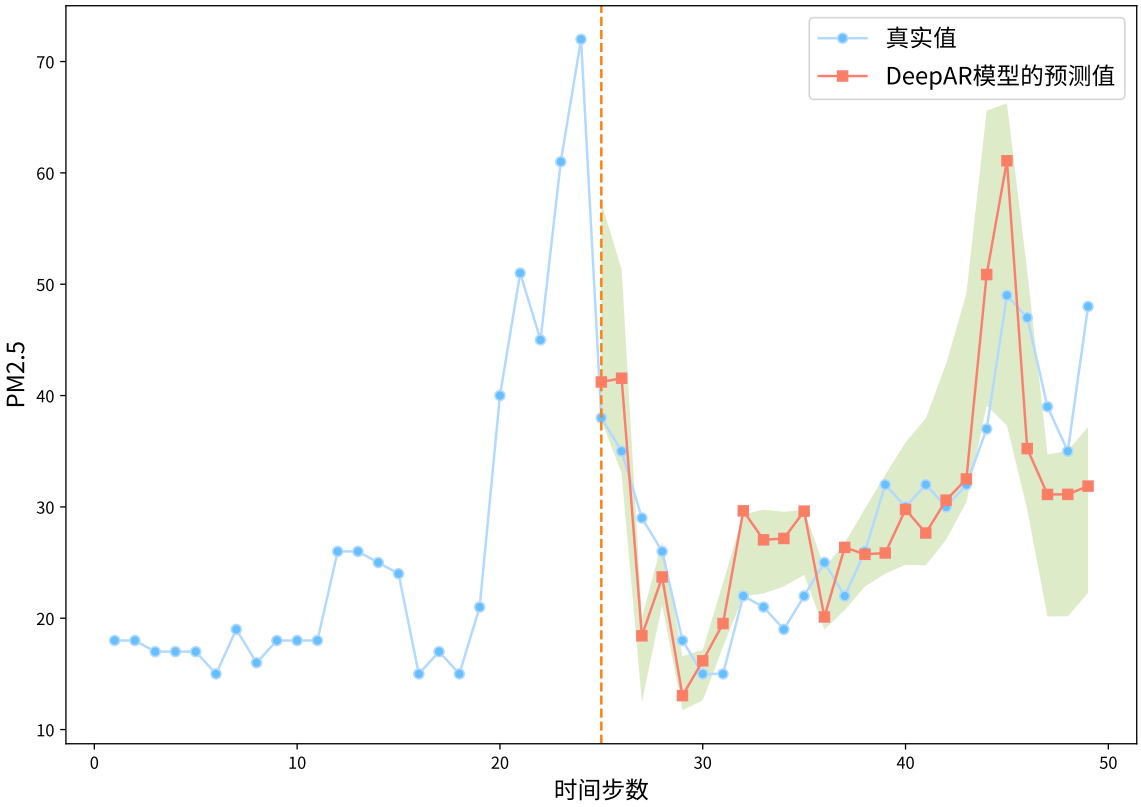


图 3-14 原始 DeepAR 模型在预测步长 $\tau = 1$ 的结果

而在真实的某上市公式的股票时间序列数据集，我们也进行单步预测。从表3-3可知当单步预测时，GCNN-DeepAR 模型比原始的 DeepAR 模型在指标

表 3-3 某上市公司股票时间序列数据集对比实验结果

模型	未来时刻 t	指标					
		MSE	RMSE	ND	ρ-risk-10	ρ-risk-50	ρ-risk-90
LSTM	$t=1$	0.05052	0.01307	0.03693	-	0.04342	-
LSTNet	$t=1$	0.04709	0.01262	0.03592	-	0.04250	-
TCN	$t=1$	0.04544	0.01239	0.03555	-	0.04183	-
DeepAR	$t=1$	0.06183	0.01446	0.03350	0.06432	0.04208	0.01791
GCNN-DeepAR	$t=1$	0.02060	0.00834	0.02369	0.06400	0.03720	0.01105

RMSE、ND、ρ-risk-10、ρ-risk-50、ρ-risk-90 都有所提高。为了更为直观地描绘 GCNN-DeepAR 模型和原始的 DeepAR 模型的预测，图3-15表示 GCNN-DeepAR 模型在预测步长 $\tau=1$ 的结果，其中浅绿色的上下界为 $[\mu-2*\sigma,\mu+2*\sigma]$ 表出，μ 和 σ 是模型预测出下一步概率分布的参数。图3-16表示原始的 DeepAR 模型在预测步

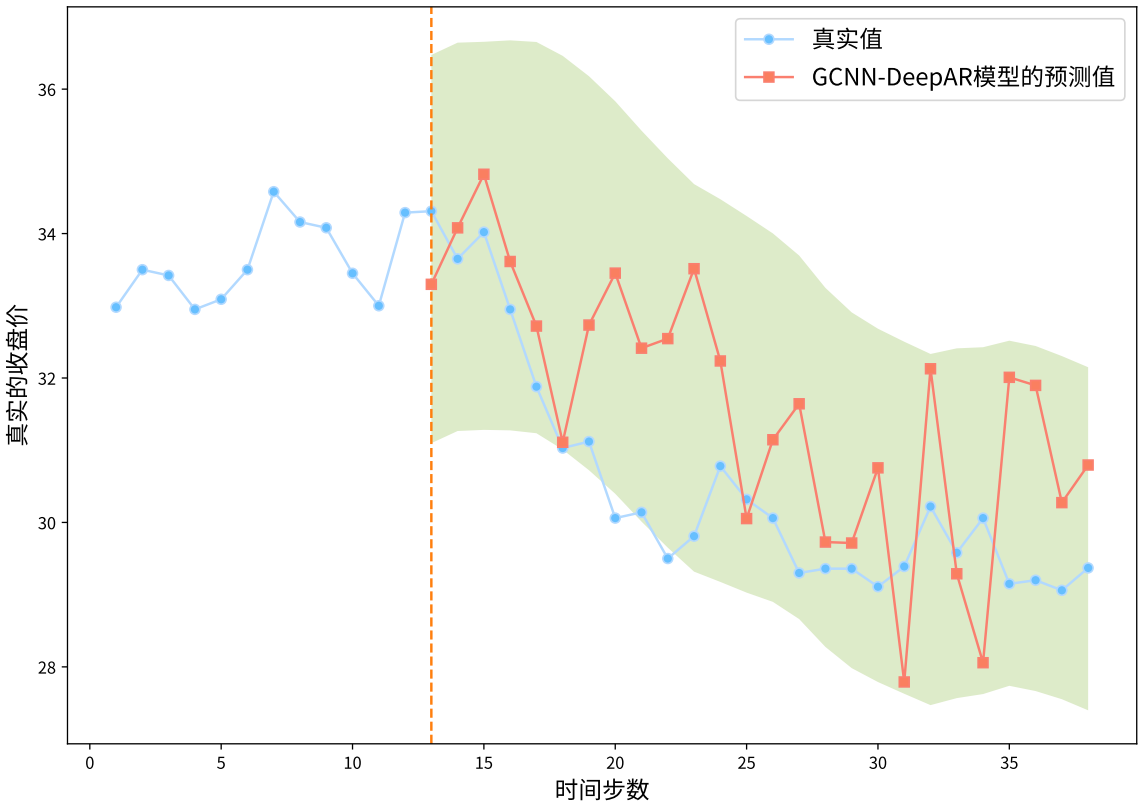


图 3-15 GCNN-DeepAR 模型在预测步长 $\tau=1$ 的结果

长 $\tau=1$ 的结果。根据表3-2和图3-15-图3-16可知，GCNN-DeepAR 模型在 RMSE，ND，ρ-risk-10，ρ-risk-50，ρ-risk-90 等指标都有提高，证明了 GCNN-DeepAR 模型

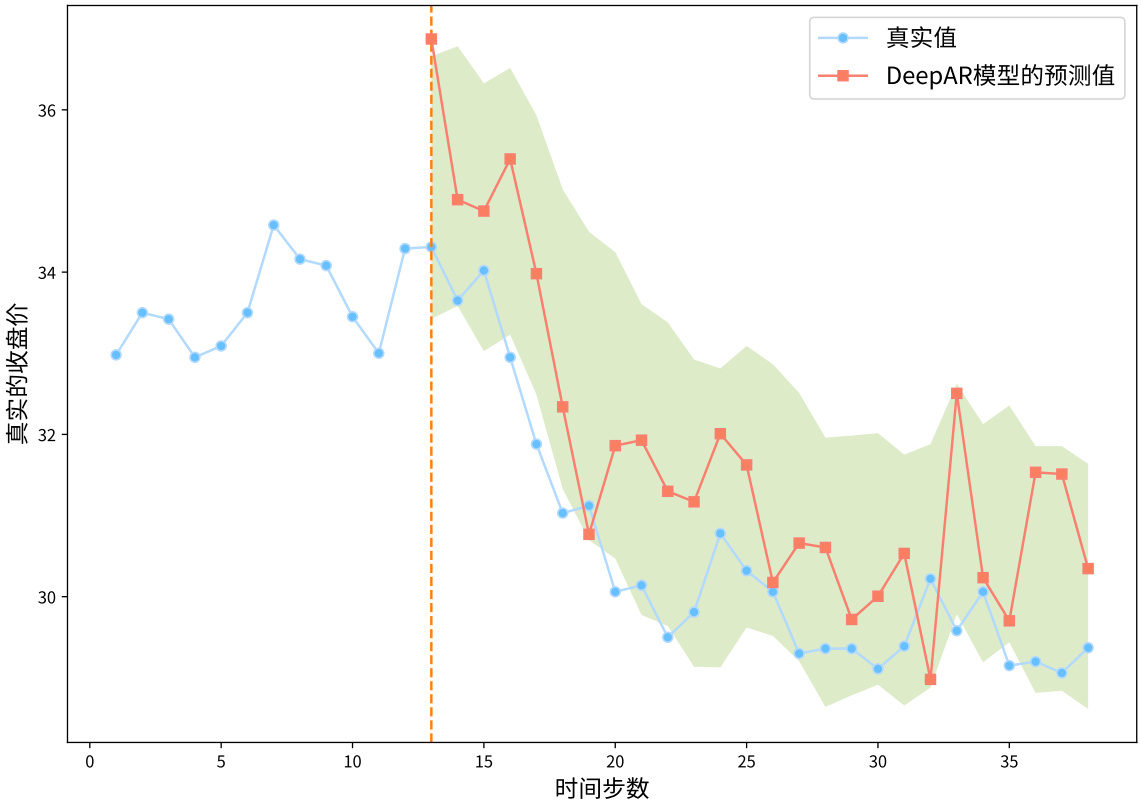


图 3-16 原始的 DeepAR 模型在预测步长 $\tau = 1$ 的结果

在真实的场景中单步预测能力也有一定的提高。

3.3.5 消融实验

为了对比 GCNN-DeepAR 单步概率预测模型的各个部分对预测的影响，我们在空气污染 PM2.5 数据集上进行消融实验。

1. 去掉 GCNN-DeepAR 模型中的一维门控卷积层部分，只保留多层 LSTM 循环神经网络和时间独立分布参数层。形成新的预测模型 Model1。

2. 去掉 GCNN-DeepAR 模型中的时间独立参数层，使用时间共享参数层。形成新的预测模型 Model2。

3. 去掉一维门控卷积层、时间独立分布参数层。完全的 DeepAR 模型。

4. 完全的 GCNN-DeepAR 模型，保留一维门控卷积层，多层 LSTM 循环神经网络，时间对立分布参数层。

实验结果如表3-4所示，我们可以看出增加了门控卷积层后，模型在多个指标有所提高。同样在增加时间独立层后，模型也在多个指标上有所提高。

表 3-4 空气污染 PM2.5 数据集消融对比实验结果

模型	指标					
	MSE	RMSE	ND	ρ-risk-10	ρ-risk-50	ρ-risk-90
Model1(去掉门控卷积层)	3.51014	0.06868	0.22894	0.22294	0.24460	0.11765
Model2(去掉时间独立层)	4.20614	0.07518	0.23623	0.18728	0.25243	0.15059
DeepAR	5.91764	0.08918	0.29670	0.17499	0.31211	0.12460
GCNN-DeepAR	2.01320	0.05201	0.19448	0.19690	0.21859	0.11902

3.4 本章小结

本章主要介绍了 GCNN-DeepAR 多变量单步概率预测模型，利用分布概率损失函数训练模型，并用训练好的模型给出预测区间。该模型使用一维门控卷积来增强模型的短周期捕获能力和控制短周期依赖信息的流动、LSTM 长短记忆神经网络来捕获全局依赖、时间独立分布参数层来增强模型的分布概括能力，让模型有更好的表征能力。与 LSTM 长短记忆神经网络、LSTNet 、TCN 时间卷积神经网络模型、原始的 DeepAR 自回归时间序列预测神经模型做对比实验，在空气污染 PM2.5 时间序列数据集和某上市公司股票时间序列数据集上，验证了 MSE、RMSE、ND、ρ-risk-10、ρ-risk-50、ρ-risk-90 等指标数据，证明了 GCNN-DeepAR 模型具有很大的表征能力，在单步预测的能力突出。

第四章 基于神经网络的多变量多步概率预测模型

传统的多步预测方法是通过单步预测模型，预测出下一时刻，通过不断地迭代预测，给出最终的多步预测结果。由于时间序列预测本身存在预测误差，在使用迭代预测过程中，预测的误差会不断地累计，最后导致模型多步预测的精度变得比较差。这种单步迭代的方法只适用于单变量多步预测问题，不太适合多变量多步的预测问题；传统的多变量多步预测模型都是给出点预测，没法量化预测的不确定性，不太适用于决策辅助。针对上述问题，本章提出了基于神经网络的多变量多步 Transformer-AR 概率预测模型，该模型是由 Masked Transformer 神经网络[23] 的编码器部分和一维门控卷积神经网络组成。在该模型上，实现一个直接预测三步的时间序列预测模型，利用分位数损失函数进行学习预测，最后给出区间预测值。

4.1 多变量多步概率预测问题描述

在时间序列任务中，我们总是通过历史的数据和掌握的先验知识来预测未来时刻的数据。有时侯，我们需要一次性预测出未来几个时刻的数据。假如我们有一段多变量历史时间序列数据 $\boldsymbol{h}=[\begin{bmatrix}h_1\\z_1\end{bmatrix},\begin{bmatrix}h_2\\z_2\end{bmatrix},\cdots,\begin{bmatrix}h_{\mathrm{T}}\\z_{\mathrm{T}}\end{bmatrix}]$，其中 $h\in\mathbb{R}^{f_1}$，$z\in\mathbb{R}$，T 是历史时间序列的长度，f_1 是时间序列数据的特征变量数，z 是预测的目标值。同时我们也有先验的知识 $\boldsymbol{x}=[x_1,x_2,\cdots,x_{\mathrm{T}+\tau}]$，其中 $x\in\mathbb{R}^{f_2}$，f_2 是先验知识的特征维数。我们希望通过模型直接预测未来 τ 个时刻的数据 $\hat{\boldsymbol{z}}=[\hat{z}_{\mathrm{T}+1},\cdots,\hat{z}_{\mathrm{T}+\tau}]$，其中 $\hat{z}\in\mathbb{R}$。这个模型可以表示为：

$$\hat{\boldsymbol{z}}=G(\boldsymbol{h},\boldsymbol{x},\Theta) \tag{4-1}$$

其中 Θ 表示是该模型的参数，模型原型如图4-1所示。

传统的方法是通过单步预测方法，每次预测出一个时刻，然后通过迭代的方式最终预测出多个时刻的数据。比如我们有一段历史时间序列数据 $\boldsymbol{h}^{(0)}=[h_1,h_2,\cdots,h_{\mathrm{T}}]$，单步预测模型 F 首先通过 $\boldsymbol{h}^{(0)}$ 预测出下一时刻 $\hat{z}_{\mathrm{T}+1}$ 的数据，然后更新模型的输入历史时间序列数据，让历史时间序列数据变为 $\boldsymbol{h}^{(1)}=[h_2,\cdots,\hat{z}_{\mathrm{T}+1}]$。然后将 $\boldsymbol{h}^{(1)}$ 作为新的输入，执行单步预测，生成下一时刻点 $\hat{z}_{\mathrm{T}+2}$ 的数据。最后不断地迭代，直到生成时刻 $\mathrm{T}+\tau$ 的预测数据 $\hat{z}_{\mathrm{T}+\tau}$。具体的迭代

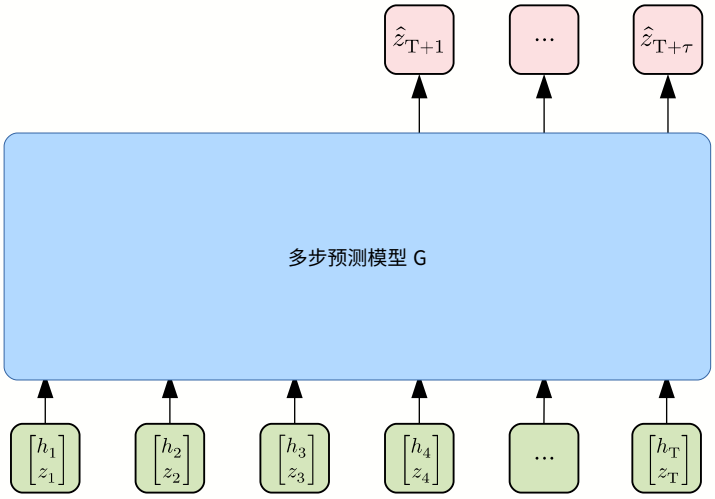


图 4-1 使用历史时间序列数据 $\boldsymbol{h} = [\begin{bmatrix} h_1 \\ z_1 \end{bmatrix}, \cdots, \begin{bmatrix} h_T \\ z_T \end{bmatrix}]$，直接进行 τ 步预测 $[\hat{z}_{T+1}, \cdots, \hat{z}_{T+\tau}]$

如公式4-2所示：

$$\begin{aligned} \hat{z}_{T+1} &= F([h_1, h_2, \cdots, h_T], \theta) \\ \hat{z}_{T+2} &= F([h_2, h_3, \cdots, \hat{z}_{T+1}], \theta) \\ &\cdots \\ \hat{z}_{T+\tau} &= F([h_t, h_{t+1} \cdots, \hat{z}_{T+\tau-1}], \theta) \end{aligned} \tag{4-2}$$

图4-2展示单步迭代预测过程，由于时间序列预测本身存在预测误差，在使用迭代预测过程中，预测的误差会不断地累计，最后导致模型多步预测的精度变得比较差。也可以看出这种单步迭代的方法也不太适应多变量的时间序列数据的场景。因此我们需要一种能通过历史时间序列数据和掌握的先验知识，适应多变量的时间序列数据，一次性的预测出多个时间点的模型。

在大部分的问题中，深度学习主要是监督学习。通过给定的标注数据，进行模型的训练和学习。为了将时间序列的预测问题引入到深度学习领域中，我们需要构造标注数据。我们将原始的时间序列数据切分为片段 $[1:T+\tau]$，其中时间片段 $[1:T]$ 的数据作为模型的输入，$[T+1, T+\tau]$ 时间序列切片数据作为模型的输出标注数据。同时，我们通过 $[1:T+\tau]$ 作为滑动窗口，对原始的时间序列数据进行采样。通过滑动窗口采样数据，这样能保留时间序列数据的时序性，让模型接触连续变化的时间序列数据，同时也能增大训练的数据集合大小，让神经网络模型更好地学习时间序列中的模式。

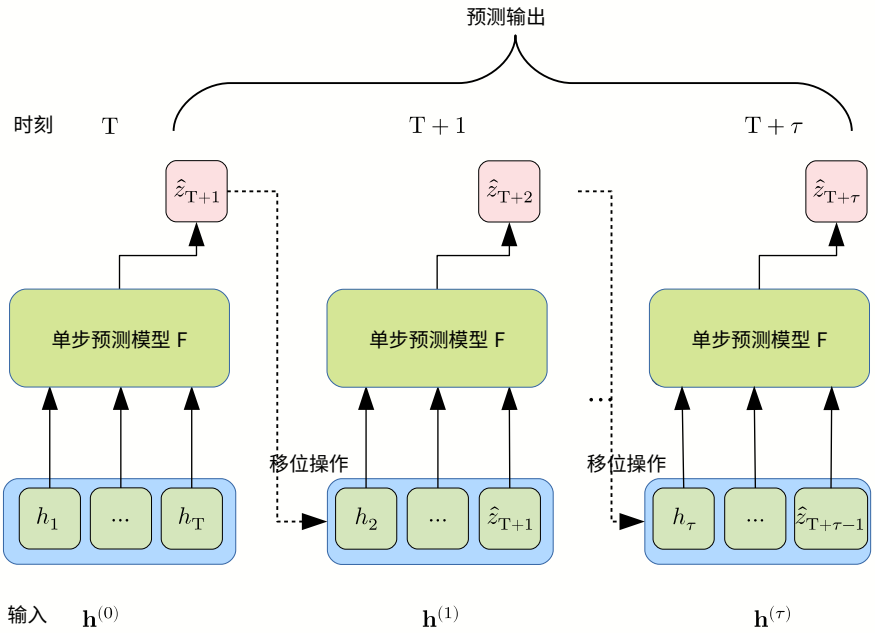


图 4-2 使用历史数据 $[h_1, \cdots, h_{\mathrm{T}}]$，迭代进行 τ 步预测 $[\hat{z}_{\mathrm{T}+1}, \cdots, h_{\mathrm{T}+\tau}]$

4.2 Transformer-AR 多步预测模型描述

在传统的循环神经网络中，很容易出现梯度的爆炸或消失，导致模型难以收敛，无法训练。尽管后来有学者提出了 LSTM[30] 长短记忆神经网络和 GRU[31] 门控循环神经网络，改善了问题，能够让循环神经网络适应于序列学习任务中，但是比较长的序列学习任务，还是会出现梯度爆炸或消失的问题，最终不能用于长序列的学习。LSTM 代表的这类循环神经网络是经过迭代运算，不能充分利用现代的硬件并行性进行计算。而 Transformer[23] 神经网络的提出，它完全依赖全局的自注意机制，不再需要迭代的运算，因此可以将其中的多头自注意力运算进行分解，让每个硬件核分别计算部分的自注意力运算。这样就可以提高运算的并行性，充分利用硬件的加速性，进而提高了训练和推理的速度。同时 Transformer 的计算复杂度为 $O(n^2 d)$，循环神经网络的计算复杂度为 $O(n \cdot d^2)$，其中 n 是输入序列的长度，d 是输入序列的数据维数。通常 d 的值是 1024 或者 512，我们可以看出在同等的算法复杂度下，Transformer 可以运用到很长的序列学习任务。而在时间序列预测问题中，如果我们的模型拥有超长的视野、能接受超长的输入时间步数和大量的历史时间序列数据，那么必然能学习到更多的时间序列数据中隐含的模式，进而得到更好的预测结果精度和更大的预测步长。就像自然语言处理领域中 BERT[39]、GPT3[40] 等无监督学习模型和在图像领域中的 ViT[41]、Swin Transformer[42] 等模

型，它们都是使用 Transformer 变种的神经网络，最后在各自领域领域取得了非常好的效果。因此，在本论文中我们使用一维门控卷积和 Masked Transformer 编码器设计一种三步概率预测的模型，用于处理多步时间预测问题。我们称作该模型为 Transformer-AR 模型，如图4-3所示。主要由输入预处理层、一维门控卷积层、Masked Transformer 编码器、三步时间独立线性层这四部分结构组成。

4.2.1 输入预处理

对于切分好的时间序列训练数据集 $\boldsymbol{h}$，由于我们的 Transformer-AR 模型的输入序列长度是有限的，如果采样窗口的数据是有尺度变化的，那么每次模型学习的时候都需要学习这个尺度变换，最终导致模型训练的不稳定性，导致很难收敛。因此我们需要对数据进行预处理，让每个采样窗口的尺度不太发生明显的变化。我们使用 DeepAR [34] 论文提出的如下经验公式：

$$\text{scale}_f = 1 + \frac{1}{\mathrm{T}} \sum_{t}^{\mathrm{T}} \boldsymbol{h}_f(t) \tag{4-3}$$

其中，$\boldsymbol{h}_f(t)$ 是指时刻 t 时时间序列中 f 特征的值。但是这个公式只有在特征都是正值时，比较管用。但是对于特征值有正有负的情况下，很容易出现 scale_f 接近于零这一特殊情况，不利于后续特征转换除法的运算，导致模型数值不稳定性。因此我们改进公式4-3让其结果总是正值，先计算训练集特征 f 的全局增加值 o_f，让其特征的所有数值 $\boldsymbol{h}_f$ 都是满足大于零的条件，通过如下公式计算：

$$o_f = \begin{cases} 0 & \text{if } \min(\boldsymbol{h}_f) \geq 0 \\ 1 + |\min(\boldsymbol{h}_f)| & \text{if } \min(\boldsymbol{h}_f) < 0 \end{cases} \tag{4-4}$$

最后改进尺度变化公式如下：

$$\text{scale}_f = 1 + \frac{1}{\mathrm{T}} \sum_{t} (\boldsymbol{h}_f(t) + o_f) \tag{4-5}$$

通过如下公式4-6，其中 $\boldsymbol{T}$ 是模型的输入序列长度。将每个数值特征值进行特征转换，然后将其转换后的时间序列输入 $\boldsymbol{h}^{(0)}$ 作为模型的输入。

$$\boldsymbol{h}_f^{(0)} = \boldsymbol{h}_f / \text{scale}_f \tag{4-6}$$

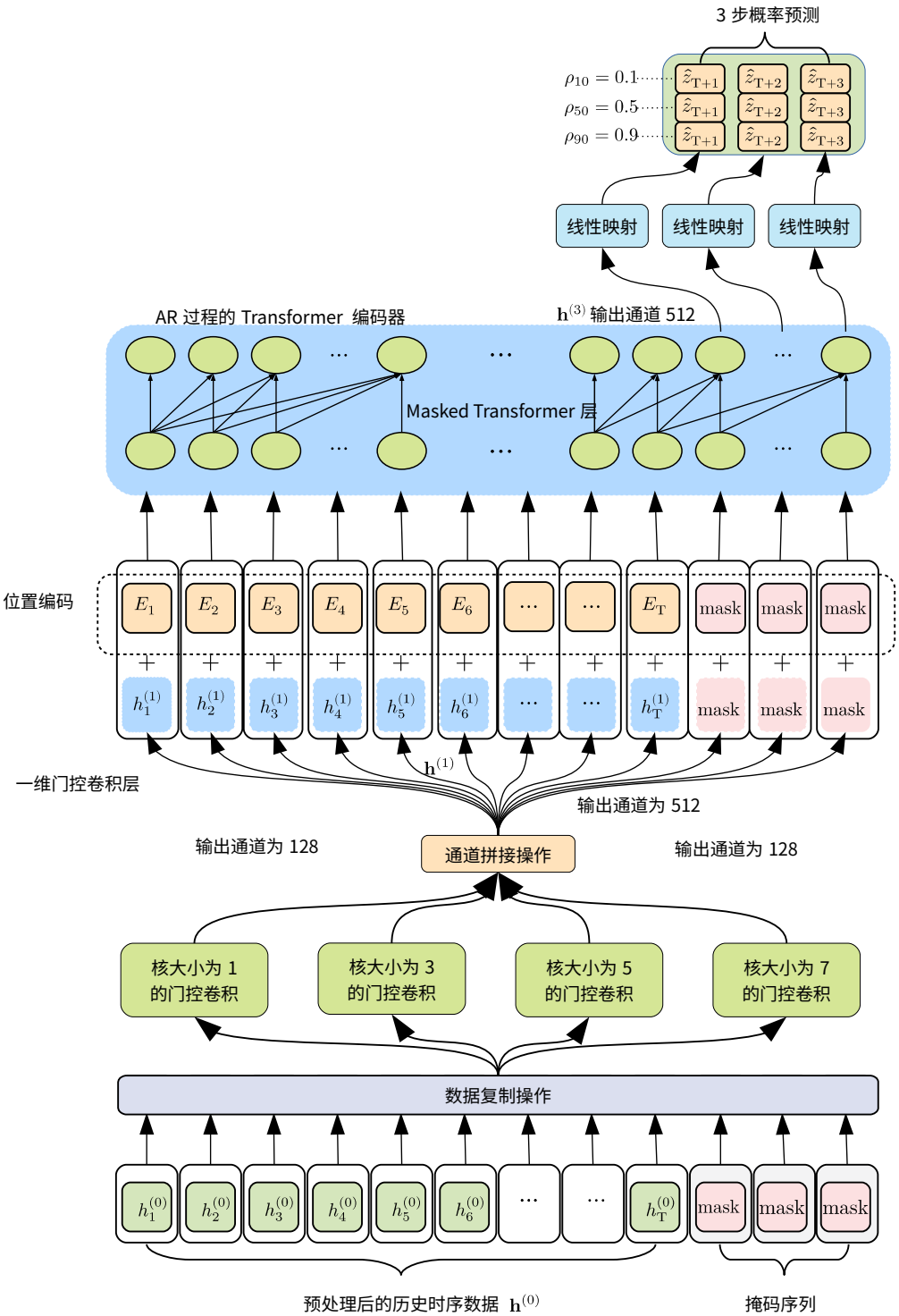


图 4-3 使用历史预处理的时间序列数据 $[h_1^{(0)},\cdots,h_T^{(0)}]$，预测未来三时刻的时间序列数据 $[\hat{z}_{T+1},\hat{z}_{T+2},\hat{z}_{T+3}]$

4.2.2 一维门控卷积层

一维门控卷积层，一维门控卷积操作一般运用在序列数据上，通过卷积操作来捕获序列的短依赖。我们已知预处理后的时间序列数据的维数为 f，我们需要我们的一维门控卷积层的输出为 d_{model}，默认我们定义为 $d_{\text{model}}=512$。我们使用 4 个一维卷积核，分别是 $(\boldsymbol{k}_1,\boldsymbol{v}_1)$，$(\boldsymbol{k}_2,\boldsymbol{v}_2)$，$(\boldsymbol{k}_3,\boldsymbol{v}_3)$，$(\boldsymbol{k}_4,\boldsymbol{v}_4)$。

其中，第一个卷积核 $(\boldsymbol{k}_1,\boldsymbol{v}_1)$ 的核大小是 1，其输入的通道数是 f，输出的通道数是 128。如图4-4所示。

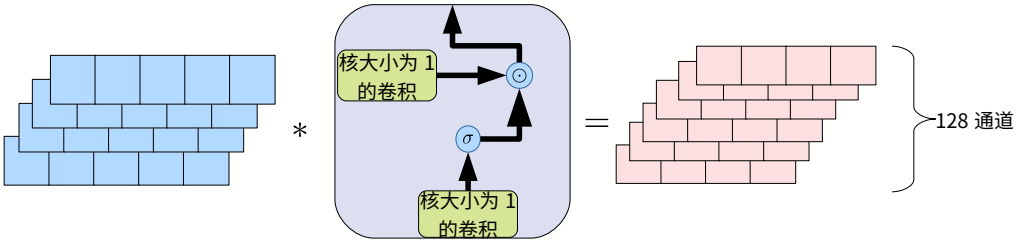


图 4-4 第一核卷积示意图

第二个卷积核 $(\boldsymbol{k}_2,\boldsymbol{v}_2)$ 的核大小是 3，其中输入的通道数是 f，输出的通道数是 128。如图4-5所示。

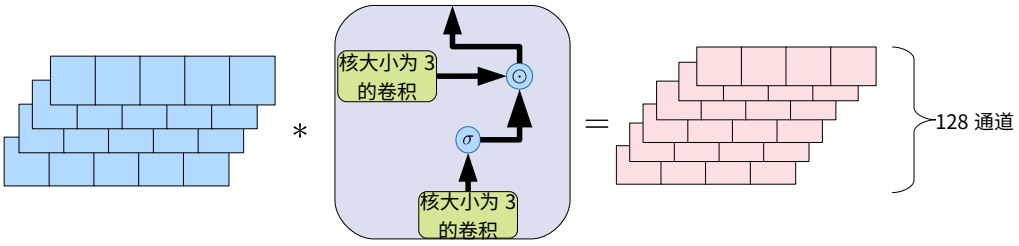


图 4-5 第二核卷积示意图

第三个卷积核 $(\boldsymbol{k}_3,\boldsymbol{v}_3)$ 的核大小是 5，其中输入的通道数是 f，输出的通道数是 128。如图4-6所示。

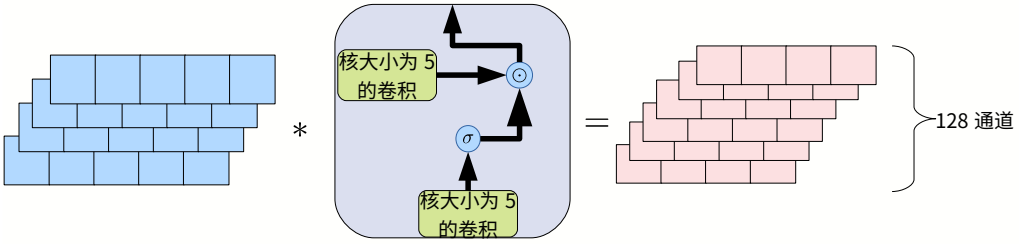


图 4-6 第三核卷积示意图

第四个卷积核 $(\boldsymbol{k}_4, \boldsymbol{v}_4)$ 的核大小是 7，其中输入的通道数是 f，输出的通道数是 128。如图4-7所示。

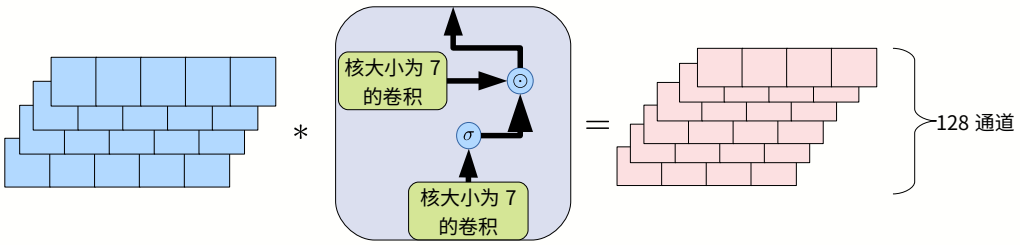


图 4-7 第四核卷积示意图

最后，我们将卷积后的数据进行通道拼接形成新的输入序列 $\boldsymbol{h}^{(1)}$，其中 $h^{(1)} \in \mathbb{R}^{512}$。整个过程如公式4-7进行运算，

$$
\begin{aligned}
\text{out}_1(i) &= \sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{k}_1(m) \odot \sigma(\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{v}_1(m)) \\
\text{out}_2(i) &= \sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{k}_2(m) \odot \sigma(\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{v}_2(m)) \\
\text{out}_3(i) &= \sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{k}_3(m) \odot \sigma(\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{v}_3(m)) \\
\text{out}_4(i) &= \sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{k}_4(m) \odot \sigma(\sum_m \boldsymbol{h}^{(0)}(m+i)\boldsymbol{v}_4(m)) \\
h^{(1)}(i) &= \text{Concat}\left(\text{out}_1(i), \text{out}_2(i), \text{out}_3(i), \text{out}_4(i)\right)
\end{aligned}
\tag{4-7}
$$

通过多个门控卷积核函数，我们尝试让模型捕获多个不同短周期的时间序列模式。

4.2.3 Masked Transformer 编码器

我们已知我们的门控卷积层的输出 $\boldsymbol{h}^{(1)}$ 是一个长度为 T 的 $d_{\text{model}} = 512$ 维数的序列。Transformer [23] 神经网络主要是由编码器模块和解码器模块组成。我们的 Transformer-AR 模型只使用 Transformer 神经网络的编码器部分。每个 transformer 块都是由一个多头注意力层、前向传播层、快捷连接 [43] 和层正则化 [44] 这四部分组成的。

由于我们使用了 Transformer 的编码器部分作为时间序列数据的长依赖捕获器，Transformer 需要进行加入位置编码，才能知道时间序列的位置信息。我们的 Transformer-AR 模型使用了自然语言预训练模型 BERT [39] 论文中提出的位置编码器，使用一个隐藏的嵌入层，该层根据神经网络的不断反馈训练生成位置编码信息 $\boldsymbol{E}$。然后将该位置编码信息和我们的一维门控卷积输出的时间序列 $\boldsymbol{h}^{(1)}$ 进行加

法运算，将位置信息加入到时间序列信息中，如公式4-8所示，

$$\boldsymbol{h}^{(1)} = [h_1^{(1)} + E_1, h_2^{(1)} + E_2, \cdots, h_{\mathrm{T}}^{(1)} + E_{\mathrm{T}}] \tag{4-8}$$

在多头注意力机制下，我们的输入序列 $\boldsymbol{h}^{(1)}$ 被完全拷贝成三分，形成 $\boldsymbol{K}$、$\boldsymbol{Q}$、$\boldsymbol{V}$。我们的 transformer 层的共有 8 个注意力头，我们定义 8 个映射矩阵 $\{\mathrm{W}_k^{(i)}\}_{i=1}^{8} \in \mathbb{R}^{512\times64}$，每个映射矩阵将原始的 $\boldsymbol{K}$ 序列映射到 64 维的空间，形成 8 个 $\{\boldsymbol{K}^{(i)}\}_{i=1}^{8}$ 的新序列；8 个转换矩阵 $\{\mathrm{W}_q^{(i)}\}_{i=1}^{8} \in \mathbb{R}^{512\times64}$，每个映射矩阵将原始的 $\boldsymbol{Q}$ 序列映射到 64 维的空间，形成 8 个 $\{\boldsymbol{Q}^{(i)}\}_{i=1}^{8}$ 的新序列；8 个映射矩阵 $\{W_v^{(i)}\}_{i=1}^{8} \in \mathbb{R}^{512\times64}$，每个映射矩阵将原始 $\boldsymbol{V}$ 序列映射到 64 维的空间，形成 8 个 $\{\boldsymbol{V}^{(i)}\}_{i=1}^{8}$ 新的序列。对每个头 ($\boldsymbol{K}_i$，$\boldsymbol{Q}_i$，$\boldsymbol{V}_i$) 我们再对其做一个自注意力运算，得到结果 $\mathrm{head}^{(i)}$。最后，将这个 8 个注意力头的输出 $\{\mathrm{head}^{(i)}\}_{i=1}^{8}$ 拼接，用一个 $\mathrm{W}_o \in \mathbb{R}^{512\times512}$ 映射矩阵对拼接的结果进行映射，形成新的输出。图4-8表示这一过程。

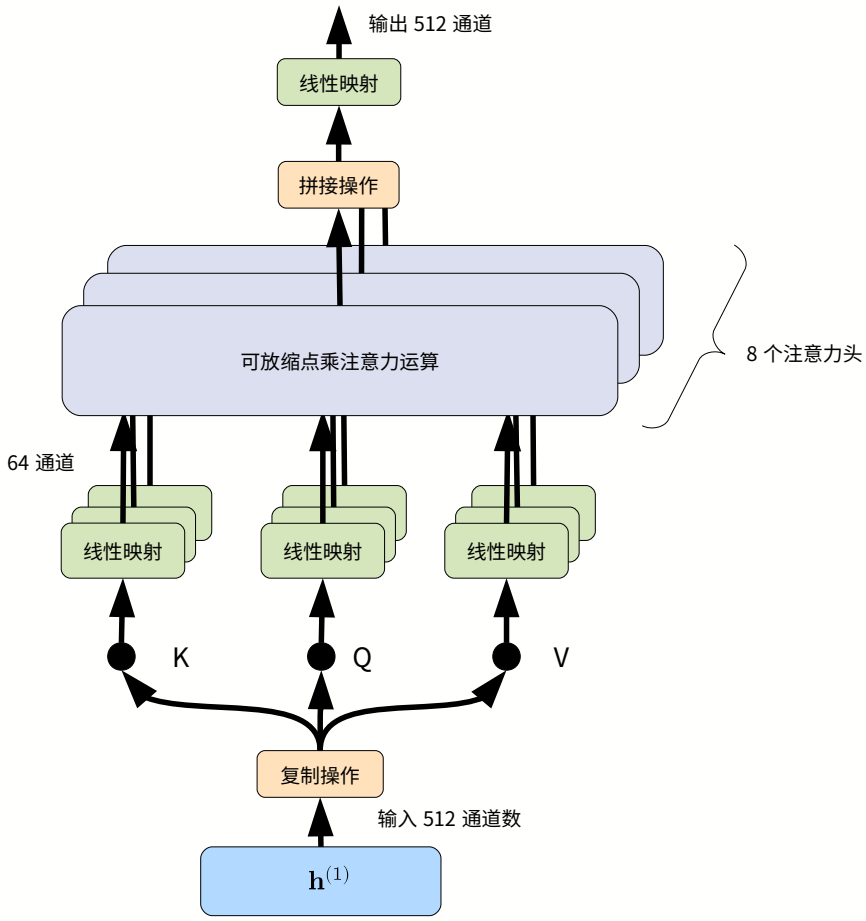


图 4-8 Transformer-AR 模型的 8 头注意力机制

我们已知引入 mask 的多头注意力运算如下，

$$\text{Attention}(\boldsymbol{K},\boldsymbol{Q},\boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_{\text{model}}}} + \text{mask}\right)\boldsymbol{V} \tag{4-9}$$

我们知道 softmax 函数在无穷区间，其值趋近于零。因此对注意力矩阵中某些元素添加一个很大的负值，让该位置的注意值约等于零，则对应的序列位置将不参与到自注意力运算中。

transformer 的注意力机制是全局的、双向的，不满足我们时间序列的时间依赖关系。因此我们需要给它增加掩码，让 Transformer-AR 模型具有因果性、单向性。定义如下的掩码方式 lamask，

$$-e^{9}\cdot\begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{4-10}$$

我们的模型预测未来最后三个时刻的数据，我们也需要设计相应的掩码，让最后三个时刻不参与到自注意运算中，定义如下的掩码方式 pmask，

$$-e^{9}\cdot\begin{bmatrix} 0 & 0 & \cdots & 0 & 1 & 1 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 \end{bmatrix} \tag{4-11}$$

最后，为了满足上述的两个约束。我们定义最终的掩码方式 mask，

$$\text{mask} = \max(\text{lamask}, \text{pmask}) \tag{4-12}$$

将这个掩码 mask 加入到自注意力运算中，图4-9表示这一过程。

4.2.4 三步时间独立线性层

Masked Transformer 层输出的序列为 $\boldsymbol{h}^{(2)} = [h_1^{(2)}, h_2^{(2)}, \cdots, h_{\mathrm{T}+3}^{(2)}]$，三步时间独立线性层有三个矩阵分别为 $\{\boldsymbol{W}^{(i)} \in \mathbb{R}^{512\times 3}\}_{i=1}^{3}$，截取 $\boldsymbol{h}^{(2)}$ 后面三个输出，得到新的输入 $\boldsymbol{h}^{(3)} = [h_{\mathrm{T}+1}^{(2)}, h_{\mathrm{T}+2}^{(2)}, h_{\mathrm{T}+3}^{(2)}]$。然后按照时间顺序和矩阵 $\{\boldsymbol{W}^{(i)}\}_{i=1}^{3}$ 进行运算，生成

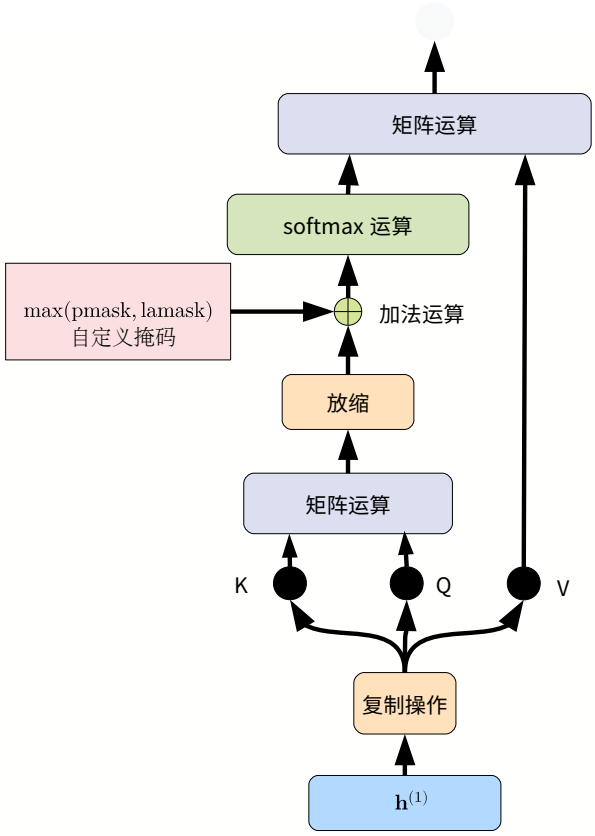


图 4-9 Transformer-AR 模型的掩码注意力机制

预测输出 $\hat{\boldsymbol{z}} = [\hat{z}_{\mathrm{T}+1}, \hat{z}_{\mathrm{T}+2}, \hat{z}_{\mathrm{T}+3}]$，其中 $\hat{z}_i \in \mathbb{R}^3$。具体过程如公式4-13所示，

$$\begin{aligned} \hat{z}_{\mathrm{T}+1} &= \mathrm{linear}(h^{(2)}_{\mathrm{T}+1}, \boldsymbol{W}^{(1)}) \\ \hat{z}_{\mathrm{T}+2} &= \mathrm{linear}(h^{(2)}_{\mathrm{T}+2}, \boldsymbol{W}^{(2)}) \\ \hat{z}_{\mathrm{T}+3} &= \mathrm{linear}(h^{(2)}_{\mathrm{T}+3}, \boldsymbol{W}^{(3)}) \end{aligned} \tag{4-13}$$

4.2.5 训练和推理过程

在训练时，我们使用分位数损失函数（Quantile loss [45]），分位数损失函数是一种无参数估计方法，通过对损失函数值进行最优化计算，来获得对应分位数的预测结果。在深度学习下的 ρ 分位数的损失函数为：

$$L(z, \hat{z}, \rho) = \frac{1}{\mathrm{T}} \sum_{i=1}^{\mathrm{T}} \{(\hat{z}_i - z_i)_+ \cdot (\rho - 1) + (z_i - \hat{z}_i)_+ \cdot \rho\} \tag{4-14}$$

其中，z_i 是真实值，$\hat{z}_i$ 是预测值。$(\cdot)_+$ 是 $\max(\cdot, 0)$ 的数学的运算符号表示。本模型中，我们分别取 $\rho_{10} = 0.1$、$\rho_{50} = 0.5$、$\rho_{90} = 0.9$。分位数损失函数对时间序列的分布函数不做任何的假设，通过选择不同的分数 ρ，我们可以训练得出不同的分位数

的预测值，因此我们可以实现如图4-10的区间预测。

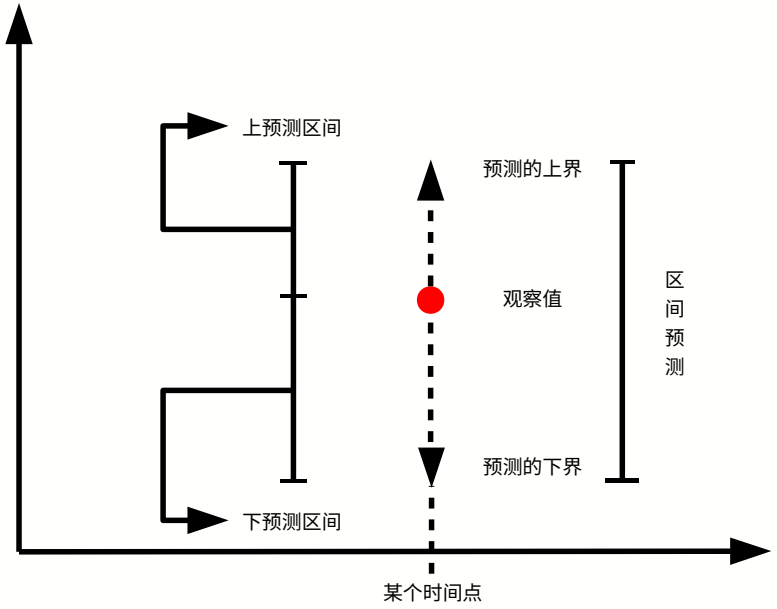


图 4-10 区间预测示意图

一个区间预测是对预测结果不确定性的量化，它由一个概率上界和概率下界组成。比如 $\rho = 0.1, \rho = 0.5, \rho = 0.9$ 时，分位数损失函数计算出的分位数值 u_1，u_2，u_3，那么我们可以确定存在如下等式成立：

$$0.1 \leq \boldsymbol{P}\{u_1 \leq Y \leq u_3\} \leq 0.9 \tag{4-15}$$

通过给出预测的上下界，我们能很好地量化预测的不确定性。

我们的模型选用 $\rho_{10} = 0.1$，$\rho_{50} = 0.5$，$\rho_{90} = 0.9$ 这三个分位数，将三步时间独立线性层的输出 $\hat{\boldsymbol{z}} = [\hat{z}_{\mathrm{T}+1}, \hat{z}_{\mathrm{T}+2}, \hat{z}_{\mathrm{T}+3}]$ 分别作为分布损失函数的输入。使用我们的滑动窗口采样的数据，用窗口中的最后三个时间点的预处理数据的目标值 $\boldsymbol{z}^{(0)}_{\mathrm{T}+1:\mathrm{T}+3}$ 作为训练的标注数据，联合 ρ_{10}，ρ_{50}，ρ_{90} 三个不同分位数损失函数对模型进行训练。最终的 Transformer-AR 模型的损失函数如下：

$$J = \sum \begin{pmatrix} L(\boldsymbol{z}^{(0)}_{\mathrm{T}+1:\mathrm{T}+3}, \hat{\boldsymbol{z}}_{1:3}[0], \rho_{10}) \\ +L(\boldsymbol{z}^{(0)}_{\mathrm{T}+1:\mathrm{T}+3}, \hat{\boldsymbol{z}}_{1:3}[1], \rho_{50}) \\ +L(\boldsymbol{z}^{(0)}_{\mathrm{T}+1:\mathrm{T}+3}, \hat{\boldsymbol{z}}_{1:3}[2], \rho_{90}) \end{pmatrix} \tag{4-16}$$

在推理中，模型通过输入历史时间序列数据 $\boldsymbol{h} = [h_1, \cdots, h_{\mathrm{T}}]$, 对其做公式4-3所示的尺度变换，模型输出预测的三个时间点数据 $\hat{\boldsymbol{z}} = [\hat{z}_{\mathrm{T}+1}, \hat{z}_{\mathrm{T}+2}, \hat{z}_{\mathrm{T}+3}]$, 我们也需要

进行反转换，通过公式4-17将预测值 $\hat{z}$ 调整为真实的预测值。

$$\hat{z}_t[1:3] = \hat{z}_t[1:3] \cdot \text{scale}_t - o_t \tag{4-17}$$

其中，o_t 是按照公式4-4计算的特征目标值全局增加量。scale_t 是按照公式4-6计算的特征目标值的尺度变换值。

4.3 实验过程与结果

4.3.1 实验内容介绍

实验软硬环境：本章实验的硬件环境为包括 Intel Xeon E5 处理器，NVIDIA GeForce RTX 3090 24G 显存，256G 内存的台式服务器。本章的所使用的软件环境为：NVIDIA-cuda 11.4，python 3.7.11，深度学习框架 pytorch-GPU 1.10.0，机器学习框架 sklearn 1.0.2，数据处理框架 pandas 1.3.5。操作系统为 18.04.6 LTS，内核版本为 5.11.0-41-generic x86_64。

训练参数：本文中提出的 Transformer-AR 模型需要调节的参数，如表4-1。

表 4-1 参数设置

模型参数	数值形式
训练批次 (batch size)	100
数据窗口 (window size)	200/600
初始化学习率	0.001
优化方法	adam
CNN 卷积核	[1, 3, 5, 7]
CNN 卷积输出维数	[64, 64, 64, 64]
CNN 卷积核边界填充	0
Transformer 层数	2
Transformer 头数	8
Transformer 输入/输出维数	512/512
预测的未来时刻	[1, 2, 3]
训练集比例	0.8
测试集比例	0.1
验证集比例	0.1

实验数据：本实验主要使用两个数据集，电力消耗数据集 [46] 和某奶场生化指标数据集。其中电力数据集从 2017 年 1 月 1 日 0 点 0 分到 2017 年 3 月 11 日 10 点 30 分每 10 分钟记录的用电量，共 10000 条数据。包括用电量、温度、湿度、风速、扩散流、一般扩散流参数。图4-11是其后 805 时间步的电力消耗数据。某奶场生化指标数据集从 2017 年 4 月 5 日到 2022 年 3 月 22 日每天的生化指标记录量，包括

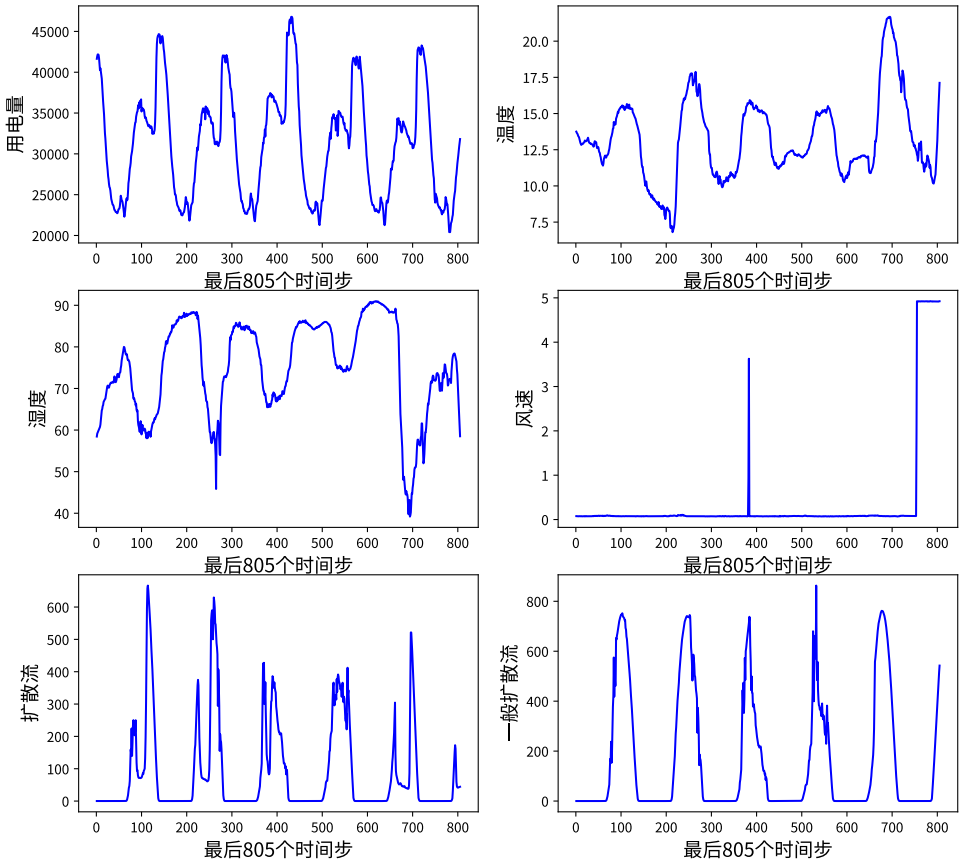


图 4-11 后 805 时间步的电力消耗数据

蛋白质、体细胞、乳糖、脂肪等参数，我们对其中缺失的值，进行三阶线性插值填充后，共有 1083 条数据。图4-12是奶场生化指标数据时间序列图。模型评估指标：本章使用常见的评估指标 MSE，RMSE，ND，ρ-risk。假定有一段真实的时间序列 $z_{i,t}$，Transformer-AR 模型预测的中位数为 $\hat{z}_{i,t}$。

MSE 定义为：

$$\mathrm{MSE} = \frac{\sum_{i,t}\left(z_{i,t} - \hat{z}_{i,t}\right)^2}{\sum_{i,t}\left|z_{i,t}\right|} \tag{4-18}$$

RMSE 定义为：

$$\mathrm{RMSE} = \frac{\sqrt{\frac{1}{\mathrm{T}}\sum_{i,t}\left(z_{i,t} - \hat{z}_{i,t}\right)^2}}{\frac{1}{\mathrm{T}}\left|\sum_{i,t} z_{i,t}\right|} \tag{4-19}$$

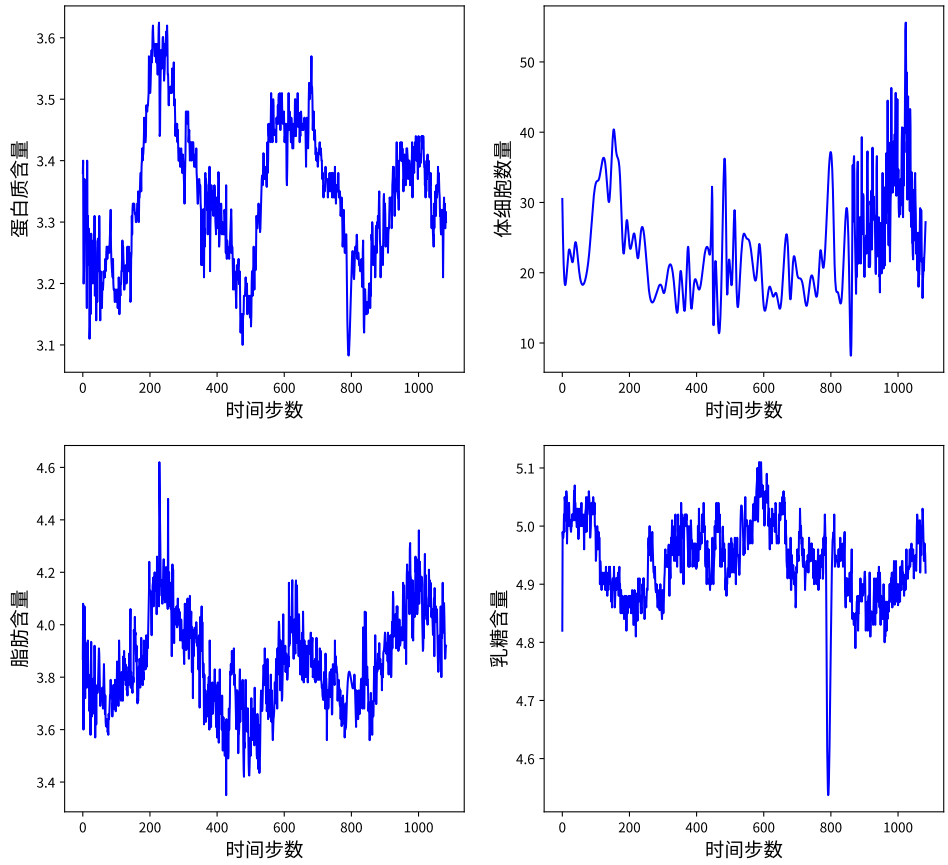


图 4-12 奶场生化指标数据时序图

ND 定义为：

$$\mathrm{ND} = \frac{\sum_{i,t} |z_{i,t} - \hat{z}_{i,t}|}{\sum_{i,t} |z_{i,t}|} \tag{4-20}$$

我们已知前文 quantile loss 定义如公式4-14所示，我们指定一个分数 ρ，Transformer-AR 模型对应预测的序列为 $\hat{z}^{\rho}_{i,t}$，那么 ρ-risk 定义为：

$$\rho\text{-risk} = \frac{2 * \sum_{i,t} \left\{ (\hat{z}^{\rho}_{i,t} - z_{i,t})_+ \cdot (\rho - 1) + (z_{i,t} - \hat{z}^{\rho}_{i,t})_+ \cdot \rho \right\}}{\sum_{i,t} |z_{i,t}|} \tag{4-21}$$

4.3.2 数据集分析与处理

对于电力数据集，包括用电量、温度、湿度、风速、扩散流、一般扩散流六个变量，其中用电量、温度、湿度、风速、扩散流、一般扩散流是数值变量。某奶场

生化指标数据集，包括蛋白质、体细胞、乳糖、脂肪四个变量，所有的变量都是数值变量。

1. **数据切分**

对于电力消耗数据集，我们采用滑动窗口的方法进行数据采样，其中窗口的大小为 T = 603，使用 $\boldsymbol{h}[1:600]$ 的时间序列数据切片作为模型的输入，$\boldsymbol{z}[601:603]$ 的时间序列数据切片作为模型的预测真实值。对于某奶场生化指标数据集，同样使用滑动窗口的方法进行数据采样，其中窗口的大小为 T = 203，使用 $\boldsymbol{h}[1:200]$ 的时间序列数据切片作为模型的输入，$\boldsymbol{z}[200:203]$ 的时间序列数据切片作为模型的预测真实值。

2. **数据预处理**

对于模型中的类别特征，我们简单的使用 label Encode 进行数值编码。而对于数值性特征，由于我们的 Transformer-AR 模型的输入序列长度是有限的，如果采样窗口的数据是有尺度的变化的，那么每次模型学习的时候都需要学习这个尺度变化，最终导致模型不能收敛。因此我们需要对数据进行预处理，让每个窗口的尺度不太发生变化。本实验使用了本章的经验尺度公式。

4.3.3 对比实验组

本文中我们使用四种模型进行对比：

1. GRU 门控神经网络模型：使用 Quantile loss 作为损失函数，使用本章的尺度变换公式4-6和反变换公式 4-17进行数据预处理。GRU 模型是经典的循环神经网络，能够很好的进行非线性的时间序列预测。

2. CNN-GRU 模型：使用 Quantile loss 作为损失函数，使用本章的尺度变换公式4-6和反变换公式4-17进行数据预处理，CNN-GRU 模型使用一维卷积神经网络对数据进行卷积操作，捕获时间序列的短依赖；使用 GRU 模型来捕获时间序列的长依赖。

3. Seq2Seq 模型：使用 Quantile loss 作为损失函数，使用本章的尺度变换公式4-6和反变换公式4-17进行数据预处理，Seq2Seq 使用 GRU 门控神经网络进行输入编码，同样使用 GRU 门控神经网络进行解码。

4. Transformer-AR 模型：使用 Quantile loss 作为损失函数，使用本章的尺度变换公式4-6和反变换公式4-17进行数据预处理，Transformer-AR 模型也是先使用一维门控卷积神经网络对数据进行卷积操作，捕获时间序列数据中的多个短依赖；最后通过 masked Transformer 来捕获时间序列的长依赖。

4.3.4 预测效果分析

在电力消耗数据集上，$t=1$，表示使用时间序列数据 $\boldsymbol{h}[1:600]$ 来预测真实的 $\boldsymbol{z}[601]$ 的数据；$t=2$，表示使用时间序列数据 $\boldsymbol{h}[1:600]$ 来预测真实的 $\boldsymbol{z}[601:602]$ 的数据；$t=3$，表示使用时间序列数据 $\boldsymbol{h}[1:600]$ 来预测真实的 $\boldsymbol{z}[601:603]$ 的数据。从表4-2可知：当预测步长 $\tau=1,\tau=2,\tau=3$ 时，Transformer-AR 模型比

表 4-2 电力消耗数据集上对比实验结果

模型	指标	预测的步长值 τ		
		$\tau=1$	$\tau=2$	$\tau=3$
GRU	MSE	11.23896	16.00861	22.29319
	RMSE	0.00138	0.00164	0.00194
	ND	0.01433	0.01685	0.01928
	ρ-risk-10	0.00673	0.00769	0.00896
	ρ-risk-50	0.01434	0.01687	0.01929
	ρ-risk-90	0.00762	0.00863	0.00975
CNN-GRU	MSE	13.69674	17.39941	21.54901
	RMSE	0.00152	0.00171	0.00191
	ND	0.01702	0.01907	0.02119
	ρ-risk-10	0.00808	0.00889	0.00976
	ρ-risk-50	0.01704	0.01909	0.02121
	ρ-risk-90	0.00775	0.00867	0.00936
Seq2Seq	MSE	6.38315	10.24317	15.53138
	RMSE	0.00104	0.00131	0.00162
	ND	0.01067	0.01342	0.01613
	ρ-risk-10	0.00497	0.00823	0.01033
	ρ-risk-50	0.01069	0.01344	0.01615
	ρ-risk-90	0.00547	0.00663	0.00806
Transformer-AR	MSE	5.89622	8.42064	9.74155
	RMSE	0.00100	0.00119	0.00128
	ND	0.00997	0.01164	0.01268
	ρ-risk-10	0.00500	0.00596	0.00641
	ρ-risk-50	0.00998	0.01166	0.01270
	ρ-risk-90	0.00494	0.00603	0.00663

GRU 模型、Seq2Seq 模型、CNN-GRU 模型在指标 RMSE，ND，ρ-risk-10，ρ-risk-50，ρ-risk-90 都有所提高。为了更为直观地描绘 Transformer-AR 和 CNN-GRU 模型的预测，图4-13表示 Transformer-AR 模型在预测步长 $\tau=1$ 的结果，其中浅绿部分的上界表示模型预测 $\rho_{90}=0.9$ 分位数的值，下界表示模型预测 $\rho_{10}=0.1$ 分位数的值。图4-14表示 CNN-GRU 模型在预测步长 $\tau=1$ 的结果，其中浅绿部分的上界表示模型预测 $\rho_{90}=0.9$ 分位数的值，下界表示模型预测 $\rho_{10}=0.1$ 分位数的值。

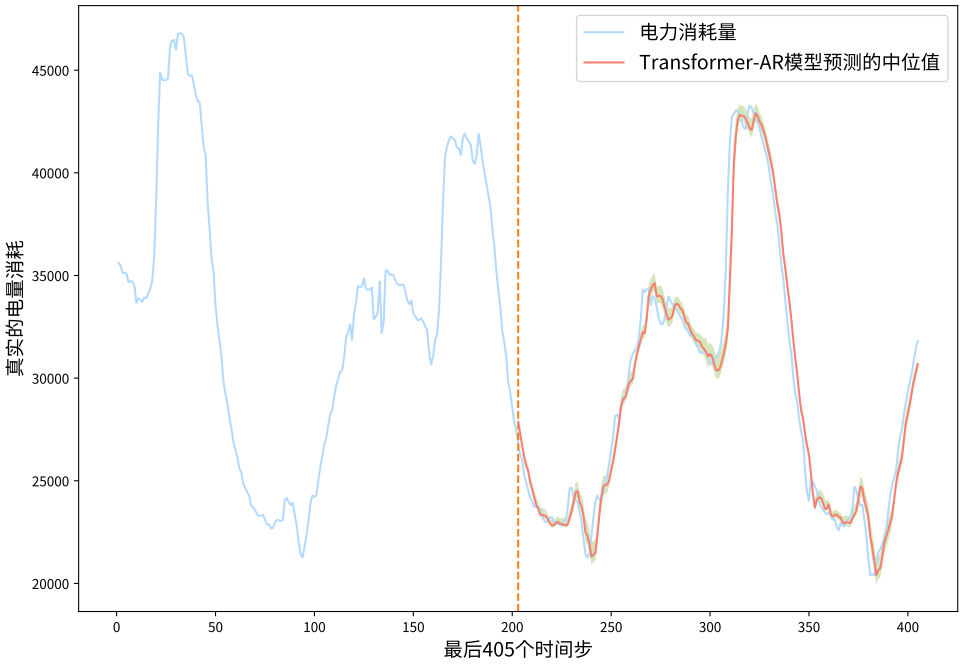


图 4-13 Transformer-AR 模型在预测 $\tau = 1$ 步的结果

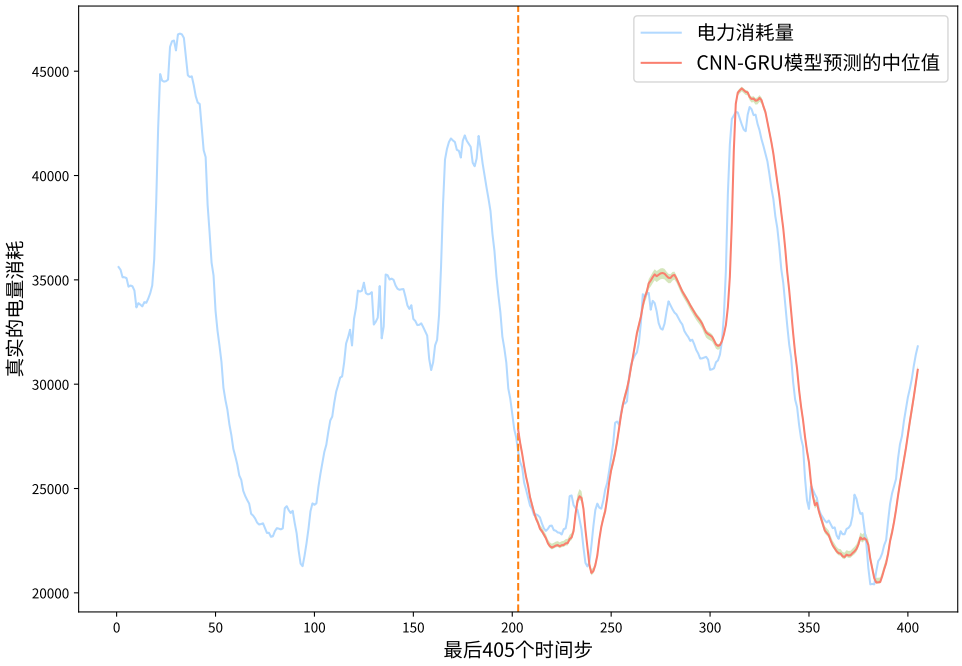


图 4-14 CNN-GRU 模型在预测 $\tau = 1$ 步的结果

在某奶场生化指标的数据集上，$\tau=1$，表示使用时间序列数据 $\boldsymbol{h}[1:200]$ 来预测真实的 $\boldsymbol{z}[201]$ 的数据；$\tau=2$，表示使用时间序列数据 $\boldsymbol{h}[1:200]$ 来预测真实的 $\boldsymbol{z}[201:202]$ 的数据；$\tau=3$，表示使用时间序列数据 $\boldsymbol{h}[1:200]$ 来预测真实的 $\boldsymbol{z}[201:203]$ 的数据。从表4-3可知：当预测步长 $\tau=1,\tau=2,\tau=3$ 时，Transformer-

表 4-3 某奶场生化指标数据集上对比实验结果

模型	指标	预测的步长值 τ		
		$\tau=1$	$\tau=2$	$\tau=3$
GRU	MSE	0.00388	0.00403	0.00468
	RMSE	0.01085	0.01107	0.01193
	ND	0.03148	0.03186	0.03442
	ρ-risk-10	0.55075	0.55104	0.55115
	ρ-risk-50	0.31416	0.31387	0.31364
	ρ-risk-90	0.06083	0.06077	0.06072
CNN-GRU	MSE	0.00212	0.00221	0.00227
	RMSE	0.00176	0.00180	0.00182
	ND	0.02317	0.02362	0.02381
	ρ-risk-10	0.54485	0.54536	0.54583
	ρ-risk-50	0.32126	0.32167	0.32182
	ρ-risk-90	0.07231	0.07288	0.07301
Seq2Seq	MSE	0.00122	0.00111	0.00110
	RMSE	0.00609	0.00582	0.00579
	ND	0.01725	0.01669	0.01661
	ρ-risk-10	0.55376	0.55350	0.55342
	ρ-risk-50	0.31800	0.31607	0.31433
	ρ-risk-90	0.06073	0.06077	0.06083
Transformer-AR	MSE	0.00087	0.00098	0.00102
	RMSE	0.00113	0.00120	0.00122
	ND	0.01292	0.01337	0.01357
	ρ-risk-10	0.53982	0.53973	0.53963
	ρ-risk-50	0.30199	0.30139	0.30116
	ρ-risk-90	0.06007	0.06033	0.06048

AR 模型比 GRU 模型、Seq2Seq 模型、CNN-GRU 模型在指标 MSE，RMSE，ND 指标有所提高，ρ-risk-10，ρ-risk-50，ρ-risk-90 指标差距不大。为了更为直观地描绘 Transformer-AR 和 CNN-GRU 模型的预测，图4-15表示 Transformer-AR 模型在预测 $\tau=1$ 的结果，图4-16表示 CNN-GRU 模型在预测 $\tau=1$ 的结果。根据表4-3和图4-15-图4-16可知，Transformer-AR 模型在 MSE，RMSE，ND 指标有所提高，证明了在实际问题中，Transformer-AR 三步概率预测模型的预测精度有很好的提高。

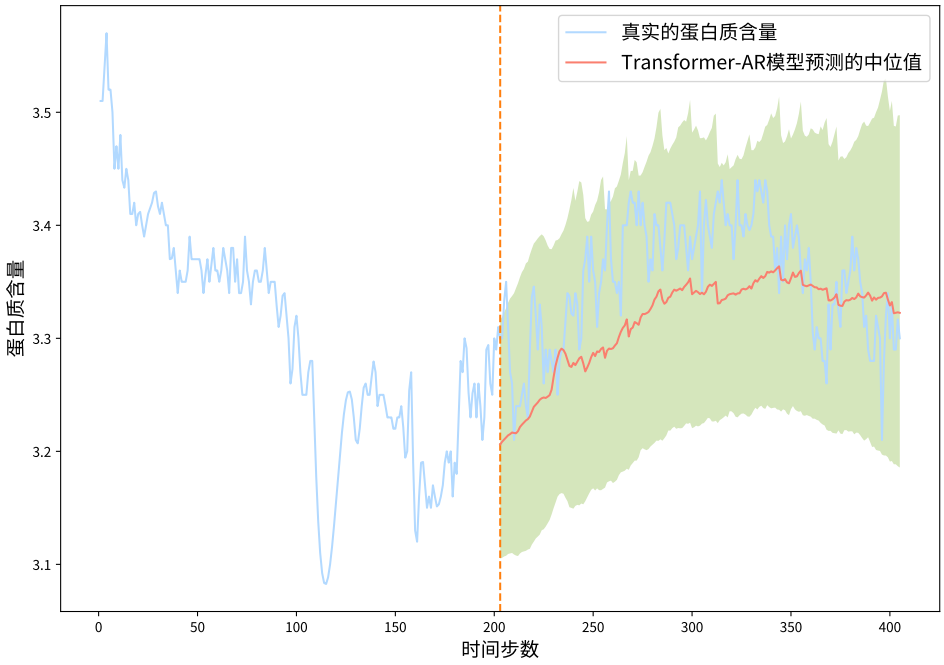


图 4-15 Transformer-AR 模型在预测 $\tau = 1$ 的结果

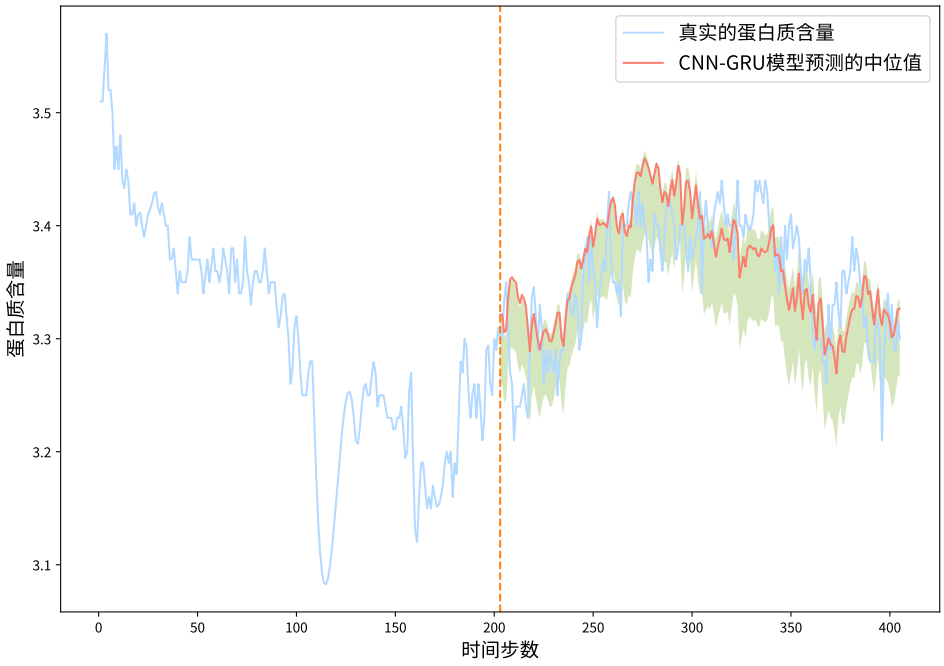


图 4-16 CNN-GRU 模型在预测 $\tau = 1$ 的结果

4.3.5 消融实验

为了对比 Transformer-AR 多步概率预测模型的各个部分对预测的影响，我们在电力消耗数据集上进行消融实验。

1. 去掉 Transformer-AR 模型中的一维门控卷积层部分，只保留 Masked-Transformer 编码器部分，形成新的预测模型 Model1。

2. 将 Masked-Transformer 编码器部分换成 GRU 循环神经网络，形成新的预测模型 Model2。

3. 完全的 Transformer-AR 模型，保留一维门控卷积层、Masked-Transformer 编码器，三步时间独立层。

实验结果如表4-4 所示，我们可以看出增加了一维门控卷积层、Masked Transformer 后，模型在多个指标上有所提高。

表 4-4 电力消耗数据集上消融对比实验结果

模型	指标	预测的步长值 τ		
		$\tau=1$	$\tau=2$	$\tau=3$
Model1(去掉一维门控卷积层)	MSE	6.15905	9.46230	13.43225
	RMSE	0.00102	0.00126	0.00150
	ND	0.01015	0.01222	0.01431
	ρ-risk-10	0.00503	0.00598	0.00681
	ρ-risk-50	0.01017	0.01224	0.01433
	ρ-risk-90	0.00628	0.00771	0.00872
Model2(将 Masked Transfomer 替换为 GRU)	MSE	15.58899	18.30531	21.24365
	RMSE	0.00162	0.00176	0.00189
	ND	0.01705	0.01868	0.02027
	ρ-risk-10	0.01399	0.01484	0.01569
	ρ-risk-50	0.01706	0.01870	0.02028
	ρ-risk-90	0.00709	0.00776	0.00830
Transformer-AR	MSE	5.89622	8.42064	9.74155
	RMSE	0.00100	0.00119	0.00128
	ND	0.00997	0.01164	0.01268
	ρ-risk-10	0.00500	0.00596	0.00641
	ρ-risk-50	0.00998	0.01166	0.01270
	ρ-risk-90	0.00494	0.00603	0.00663

4.4 本章小结

本章主要介绍了 Transformer-AR 多变量多步概率预测模型，该模型使用一维门控卷积来增强模型的短周期捕获能力、Masked Transformer 神经网络来捕获全局依赖，利用分位数损失函数训练模型，对每个预测时间点同时给出 ρ_{10}，ρ_{50}，ρ_{90}

分位数的预测值。改进了时间序列数据尺度变换和反变换公式，适应特征变量有正有负的情况。通过和 GRU 门控循环神经网络模型、CNN-GRU 神经网络模型、Seq2Seq 神经网络模型做对比实验，在电力消耗数据集和某奶场生化指标数据集，验证了 MSE、RMSE、ND、ρ-risk-10、ρ-risk-50、ρ-risk-90 等指标数据，证明了 Transformer-AR 模型具有很强的表征能力，能够高精度进行多变量多步区间预测，为商业操作提供了有依据的决策。

第五章 全文总结与展望

在本文中，我们针对单步时间序列预测问题，提出了一种 GCNN-DeepAR 时间序列概率预测模型。先对时间序列的分布进行假设（比如假设时间序列满足高斯分布、负二项分布等分布）。训练时，利用分布的最大化 log 似然函数估计来训练模型参数。推理时，预测出下一时刻 $T+1$ 的分布参数，然后利用该分布参数构造分布并进行 Monte Carlo 采样，得到最终的预测值 $\hat{z}_{T+1}$。同时利用预测出的参数，给出预测的概率上界和下界，量化预测的不确定性。

我们可以通过单步模型进行迭代来获得多步的预测结果，但是通过迭代的方式容易累加误差，并且不太适应多变量时间序列预测问题。因此，针对多变量多步预测问题，我们提出了 Transformer-AR 模型来直接进行三步时间的概率预测。使用了多个不同核的一维卷积操作，来捕获时间序列中的短周期模式；设计了满足我们三步时间预测问题的掩码，通过该掩码将 Transformer 神经网络的编码器运用到我们的三步时间预测问题中。通过使用分位数损失函数，实现对同一时刻同时给出 $\rho_{10}=0.1, \rho_{50}=0.5, \rho_{90}=0.9$ 三个分位数的预测值。给出量化的不确定性的上下界。

Transformer[23]，最近几年自然语言处理方面大放光彩。比如 GPT3[40]、BERT[39] 等自监督学习预训练模型，大大刷新了计算机自然语言处理能力。最近有学者将 Transformer 神经网络运用到图像处理领域，提出了 ViT[41]（Vision Transformer）。ViT 第一次将 Transformer 神经网络运用到图像分类的任务上，通过大量的标记数据，发现 Transformer 神经网络能比卷积神经网络取得更好的分类精度。接着针对 ViT[41] 只能适应分类问题，不适应比如图像目标检测、目标像素分割等细腻的问题，有学者提出了 SwinTransformer[42] 模型，通过数据训练模型，让 SwinTransformer 成为目标检测算法、目标像素分割等算法的图像特征提取器，最终超过了以前卷积神经网络的精度。那么是否 Transformer 神经网络的强大的表征能力和计算能力能够运用到时间序列预测的问题中？有学者提出 TFT[24] 模型，取得了很好的预测结果。但是它使用 LSTM 长度记忆网络来表示时序间的依赖，没法利用 Transformer 的多头并行性。我们针对三步时间预测问题，提出了融合多核一维卷积神经网络和 Masked Transformer 神经网络的 Transformer-AR 模型，捕获局部和全局的时间序列模式。Transformer-AR 不使用迭代的神经网络，完全依赖卷积操作和全局注意力。因此，底层的硬件可以通过并行化，加速网络的训练和推理。在实验中，我们发现 Transformer-AR 模型对时间序列数据更能拟合，如果

我们加大它的处理窗口，加大它学习的样本，可以实现更为有效的预测。

我们知道 Transformer 的自注意运算的复杂度为 $(n^2\dot{d})$，当我们使用超长的采样窗口如 $n = 100000$。这样会形成非常大的矩阵操作，占有大量的内存，导致无法计算。因此也有学者试图解决超长序列输入导致 Transformer 神经网络无法训练的问题，如有的学者提出 Linformer[47]，通过低秩矩阵逼近的方法近似计算全局注意力，让运算的复杂对降低线性级别，为这个问题提供了一种数学思路。又比如在 SwinTransformer[42] 通过将图像划分为不同的窗口，在每个单独的窗口进行注意力运算。设计了一个金字塔结构，小窗口到大窗口的一步步叠加，通过巧妙的窗口变换和掩码设计，最终大大降低了计算量，计算出了全局注意力。最后，我们也希望在时间序列预测领域中也能出现一种巧妙地解决超长距离的运算复杂度过高的方法。Transformer 神经网络的超高表征能力和计算并行度，必然能在超长视野时间序列预测中大放光彩。

5.1 全文总结

本文以深度学习在时间序列运用为研究背景，主要针对单步预测问题和多步预测问题进行了研究，针对单步预测问题，我们提出了 GCNN-DeepAR 混合神经网络模型，使用一维门控卷积神经网络来捕获不同的短周期的模式，控制信息的流动。最终 GCNN-DeepAR 模型比原始的 DeepAR 模型在两个真实的数据集提高了指标。针对多步预测问题，我们提出了 Transformer-AR 混合神经网络模型，融合了一维卷积神经网络的短周期捕获能力和 Masked Transformer 的全局捕获能力。给出了一个针对 3 步时间预测问题的模型，在两个数据集上取得相对与传统的门控循环神经网络 GRU 相对较好的指标。

5.2 后续工作展望

将深度学习运用到时间序列预测问题中，是非常有意义的。如果能够解决 Transformer 神经网络输入超长序列输入后，自注意力计算复杂度太高。通过搜集大量类似的历史时间序列数据，我们能够使用 Transformer-AR 模型得到更好的精度。后续，希望将 Linformer[47] 神经网络运用到 Transformer-AR 模型的全局注意力计算，使用更长的窗口进行训练模型，希望能得到更好的预测结果，为未来的生活带来更好的希望。

致 谢

在攻读硕士学位期间，首先衷心感谢我的导师高辉教授。每周的组会分享读论文，让我收益匪浅、不断地读文献、不断地写作；实验室的服务器，让我不计成本的实现自己的想法；导师身上的严谨作风，我学习的榜样，做一个正直的人。感谢我的父母，外婆。母亲的包容和鼓励让我任性地读研圆自己梦想，人生一场梦，回忆往昔历历在目。感谢我们的朋友们，刘磊、董海浪。我读研期间，依然这么地照顾我，友情没有随着时间打折。感谢前女友，让我觉得自己应该去追求自己想要的东西，人生苦短、及时行乐。感谢实验室的同学和学弟们，谢谢他们的帮助。最后，祝福自己，人生精彩！

参考文献

[1] G. E. Box, G. M. Jenkins. Some recent advances in forecasting and control[J]. Journal of the Royal Statistical Society. Series C (Applied Statistics), 1968, 17(2): 91-109.

[2] T. Chen, T. He, M. Benesty, et al. Xgboost: extreme gradient boosting[J]. R package version 0.4-2, 2015, 1(4): 1-4.

[3] G. Ke, Q. Meng, T. Finley, et al. Lightgbm: A highly efficient gradient boosting decision tree[J]. Advances in neural information processing systems, 2017, 30.

[4] A. V. Dorogush, A. Gulin, G. Gusev, et al. Fighting biases with dynamic boosting[J]. arXiv preprint arXiv:1706.09516, 2017.

[5] A. V. Dorogush, V. Ershov, A. Gulin. Catboost: gradient boosting with categorical features support[J]. arXiv preprint arXiv:1810.11363, 2018.

[6] A. K. Jain, J. Mao, K. M. Mohiuddin. Artificial neural networks: A tutorial[J]. Computer, 1996, 29(3): 31-44.

[7] M. Abadi, P. Barham, J. Chen, et al. {TensorFlow}: A system for {Large-Scale} machine learning[C]. 12th USENIX symposium on operating systems design and implementation (OSDI 16), 2016, 265-283.

[8] A. Paszke, S. Gross, F. Massa, et al. Pytorch: An imperative style, high-performance deep learning library[J]. Advances in neural information processing systems, 2019, 32.

[9] C. Cortes, V. Vapnik. Support-vector networks[J]. Machine learning, 1995, 20(3): 273-297.

[10] C.-C. Chang, C.-J. Lin. Libsvm: a library for support vector machines[J]. ACM transactions on intelligent systems and technology (TIST), 2011, 2(3): 1-27.

[11] J. H. Friedman. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001, 1189-1232.

[12] D. W. Hosmer Jr, S. Lemeshow, R. X. Sturdivant. Applied logistic regression[M]. John Wiley & Sons, 2013.

[13] C. E. Rasmussen. Gaussian processes in machine learning[C]. Summer school on machine learning, 2003, 63-71.

[14] A. J. Smola, B. Schölkopf. A tutorial on support vector regression[J]. Statistics and computing, 2004, 14(3): 199-222.

[15] 张朝元, 胡光华, 徐天泽. 基于 ls-svm 的交通流量时间序列预测 [J]. 云南大学学报：自然科学版, 2004, 26(B07): 4.

[16] W.-Y. Loh. Classification and regression trees[J]. Wiley interdisciplinary reviews: data mining and knowledge discovery, 2011, 1(1): 14-23.

[17] 孙晓黎, 马超群, 朱才华. 基于 xgboost 的轨道交通短时客流预测精度分析 [J]. 交通科技与经济, 2021.

[18] R. Frigola, F. Lindsten, T. B. Schön, et al. Bayesian inference and learning in gaussian process state-space models with particle mcmc[J]. Advances in neural information processing systems, 2013, 26.

[19] Y. LeCun, L. Bottou, Y. Bengio, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[20] 裴艳宇, 杨小彬, 传金平等. 一维卷积神经网络特征提取下微震能级时序预测 [J]. 工程科学学报, 2021, 43(7): 7.

[21] I. Goodfellow, Y. Bengio, A. Courville. Deep learning[M]. MIT press, 2016.

[22] 陈亿雄, 李苑, 刘小明等. 长短记忆神经网络在流行性感冒暴发预测中的应用 [J]. 江苏预防医学, 2019, 30(6): 4.

[23] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[24] B. Lim, S. Ark, N. Loeff, et al. Temporal fusion transformers for interpretable multi-horizon time series forecasting[J]. International Journal of Forecasting, 2021.

[25] G. E. Box, D. R. Cox. An analysis of transformations[J]. Journal of the Royal Statistical Society: Series B (Methodological), 1964, 26(2): 211-243.

[26] I.-K. Yeo, R. A. Johnson. A new family of power transformations to improve normality or symmetry[J]. Biometrika, 2000, 87(4): 954-959.

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

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

[29] A. Van Den Oord, S. Dieleman, H. Zen, et al. Wavenet: A generative model for raw audio.[J]. SSW, 2016, 125: 2.

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

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

[32] I. Sutskever, O. Vinyals, Q. V. Le. Sequence to sequence learning with neural networks[J]. Advances in neural information processing systems, 2014, 27.

[33] D. Bahdanau, K. Cho, Y. Bengio. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.

[34] V. Flunkert, D. Salinas, J. Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks[J]. International Journal of Forecasting, 2020, 36(3).

[35] Y. N. Dauphin, A. Fan, M. Auli, et al. Language modeling with gated convolutional networks[C]. International conference on machine learning, 2017, 933-941.

[36] X. Liang, T. Zou, B. Guo, et al. Assessing beijing's pm2. 5 pollution: severity, weather impact, apec and winter heating[J]. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2015, 471(2182): 20150257.

[37] G. Lai, W.-C. Chang, Y. Yang, et al. Modeling long-and short-term temporal patterns with deep neural networks[C]. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, 95-104.

[38] S. Bai, J. Z. Kolter, V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling[J]. arXiv preprint arXiv:1803.01271, 2018.

[39] J. Devlin, M.-W. Chang, K. Lee, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[40] T. Brown, B. Mann, N. Ryder, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877-1901.

[41] A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.

[42] Z. Liu, Y. Lin, Y. Cao, et al. Swin transformer: Hierarchical vision transformer using shifted windows[C]. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, 10012-10022.

[43] K. He, X. Zhang, S. Ren, et al. Deep residual learning for image recognition[C]. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, 770-778.

[44] J. L. Ba, J. R. Kiros, G. E. Hinton. Layer normalization[J]. arXiv preprint arXiv:1607.06450, 2016.

[45] R. Koenker, G. Bassett Jr. Regression quantiles[J]. Econometrica: journal of the Econometric Society, 1978, 33-50.

[46] A. Salam, A. El Hibaoui. Comparison of machine learning algorithms for the power consumption prediction:-case study of tetouan city–[C]. 2018 6th International Renewable and Sustainable Energy Conference (IRSEC), 2018, 1-5.

[47] S. Wang, B. Z. Li, M. Khabsa, et al. Linformer: Self-attention with linear complexity[J]. arXiv preprint arXiv:2006.04768, 2020.