UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Deep Learning-based Time Series Forecasting: Models and Applications

by Bin Wang

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of
Associate Professor Guangquan Zhang
and Distinguished Professor Jie Lu

University of Technology Sydney
Faculty of Engineering and Information Technology
Australian Artificial Intelligence Institute

September 2022

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Bin Wang*, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science at the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with Southwest Jiaotong University.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior to publication.

Date: 9, September, 2022

Acknowledgements

It is an extremely remarkable and unforgettable journey for pursuing my Ph.D. degree at the University of Technology Sydney. I am indebted to the people for all their help and support.

I would like to express sincere gratitude to my principal supervisor, A./Professor Guangquan Zhang. Without his encouragement and advice, I would not have been able to accomplish this Ph.D program. His vision and wisdom influence me deeply, which point out the directions of my research. It always warms me to be invited as a member of his home party. I also would like to express thankfulness to my co-supervisor, Distinguished Professor Jie Lu. Her optimism, enthusiasm and confidence guided me to the right direction, especially when I become puzzled in my research progress. She initiated the dual doctoral degree program between UTS and SWJTU, provided me scholarship and offered me an opportunity to join DeSI Lab in the Australian Artificial Intelligence Institute. In addition, I would like to express my respect to my supervisor at SWJTU, Professor Tianrui Li. His encouragement and high standards of academic pursuit motivated me to face challenges positively. It is fortunate to be his postgraduate in my life.

I would like to thank every member of the DeSI Lab. It was a wonderful experience to learn from these dedicated researchers. I especially thank Dr. Dianshuang Wu, Dr. Anjin Liu, Dr. Hang Yu, Dr. Ximeng Wang, Dr. Ruiping Yin,

Dr. Qian Zhang, Dr. Yiliao Song, Dr. Shan Xue, Dr. Feng Liu, Dr. Feng Gu and Dr. Zhen Fang who have shared their opinions and comments with me. Furthermore, I want to thank for the kind help from Dr. Mingshan Jia, Dr. Yan Zhang, Dr. Mengxi Jiang, Dr. Joakim Skarding and other friends, may our friendship last forever.

I would like to express my heartfelt appreciation to my parents for their love and support.

Abstract

With the advent of the era of big data and artificial intelligence, people can obtain various data and extract valuable information and knowledge through artificial intelligence technology. As a typical representative of complex data, time series modeling and forecasting have always been hot topics. In the big data environment, time series often have the characteristics of multi-source complexity, dynamic heterogeneity, uncertainty, and nonlinearity, which brings tremendous challenges to the processing and prediction of time series data. As a cutting-edge approach of artificial intelligence, deep learning has efficient automatic feature extraction and robust representation learning capabilities. Using deep learning to enhance time series forecasting performance has become an important research direction. This dissertation studies the point estimation and uncertainty quantification of time series forecasting based on deep learning methodology. This thesis achieves research contributions as follows:

(1) It proposes a point estimation network based on noisy residual connection and snapshot ensemble forecasting methods. The model reasonably simplifies the input data and introduces Gaussian noise to the residual connection to achieve regularization so that the model can significantly reduce the training time while ensuring generalization accuracy. The method of forecasting using snapshot ensemble for point estimation has also been explored. It can significantly

improve forecasting accuracy without much difference from the training time of a single model. The experiment is evaluated based on two realistic traffic flow datasets and confirms the effectiveness of the proposed methods.

(2) It develops a deep uncertainty quantification method based on the assumption of Gaussian distribution. The method employs a new likelihood loss function based on the Gaussian distribution as the training loss function of the encoder-decoder model, allowing the model to simultaneously predict the point estimation and prediction interval of multiple variables at multiple time steps. The research also reported the generalization improvement introduced by training the deep forecasting model with the likelihood loss function. This research applies the proposed method to solve a real-world weather forecasting problem. By designing an effective multi-source information fusion mechanism, it can improve the forecasting accuracy of various meteorological variables significantly.

(3) It further presents a deep uncertainty quantification method without distribution assumption. The method constructs a novel loss function without a distribution hypothesis, hence the so-called distribution-free loss function, as the objective loss function for training the encoder-decoder to model the data distribution adaptively at the training stage and then predict the point estimation and the prediction interval at the forecasting stage. The effectiveness of the proposed method is verified on three public datasets.

(4) It constructs a novel deep quantile fusion network for robust point estimation. By designing a novel loss function, the network can transform the hidden layer into a quantile layer with semantic information and take these quantiles as

the input features of the following layer to predict the target variable, thereby constructing a deep forecasting model with high accuracy and great robustness. The experiments are evaluated on eight UCI regression datasets and a time series dataset and demonstrate the effectiveness of the proposed methods.

This study focuses on time series data with characteristics of complexity, heterogeneity, abnormality, nonlinearity, and uncertainty. It discusses data preprocessing and feature fusion, proposes effective deep point estimation and uncertainty quantification methods, and applies these methods to handle practical forecasting tasks. The research works of this dissertation can promote the related research of deep learning in time series forecasting, promote the development of related industrial applications, and have both theoretical significance and application value.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With the rapid development of sensing technology and storage capacity, massive data are collected and saved in real-time and revolutionize modern life in many ways (Doctorow and Cory, 2008). Big data may be gathered from various fields, such as meteorology, traffic flow, financial market, etc., and have become an important asset to promote social development. Driven by the ever-increasing amount of data, supercomputing power, and emerging machine learning methods, artificial intelligence has ushered in a new wave of upsurge, and a new era of intelligence is coming (Bundy, 2017). In the parallel age of big data and artificial intelligence, embracing the opportunities and challenges and utilizing artificial intelligence technology to analyze and exploit big data has attracted extensive attention from academia and industry. Among them, time series analysis is one of the critical research areas.

A time series is a sequence of data points arranged in the order of time. Usually, the time interval of a set of time series is a constant time (such as 1 second, 5 minutes, 12 hours, 7 days, 1 year) (Wikipedia contributors, 2022). The research on time series mainly covers four aspects: time series visualization (Aigner et al., 2011), sequence anomaly detection (Gupta et al., 2014), sequence classification (Fawaz et al., 2019), and sequence prediction (De Gooijer and Hyndman, 2005). Among them, time series visualization is mainly to develop special visualization methods for the characteristics of different time series data, such as visualization methods for single-dimensional and multi-dimensional variables and visualization techniques for 2D and 3D visual dimensions (Aigner et al., 2011); sequence anomaly detection is a timely detection of abnormal states of time series, such as change point detection of financial sequences, abnormal traffic when websites are maliciously attacked, etc. (Gupta et al., 2014); Sequence classification is to classify a sequence after feature extraction and analysis, such as Human action recognition, audio scene classification, etc. (Fawaz et al., 2019). Among all time series-related problems, sequence forecasting is one of the most urgent and challenging tasks, and it is also the key research direction of this topic. Time series often are generated from dynamic systems, and time series forecasting [1] aims to simulate the underlying dynamical system by modeling the historical observed data sequence and then use the model to predict the sequence values in the future (Han et al., 2018).

[1] In a narrow sense, time series forecasting refers to predicting variables that vary over time. Broadly speaking, time series forecasting also involves spatio-temporal forecasting. For example, traffic flow is a type of time series data, and adjacent traffic intersections correlate geospatially. Therefore, traffic flow forecasting belongs to time series forecasting; more precisely, it belongs to spatio-temporal forecasting. This thesis regards time series forecasting in the broad sense.

1.2 Research Challenges and Methodologies

1.2.1 Research Challenges

In computer vision and natural language processing, images and text are the primary data types, and the data quality is generally considerable. The data preprocessing is usually implemented by normalization and text encoding. The target to be predicted rarely undergoes concept drift (Lu et al., 2019), so researchers can focus more on how to design the machine learning models. Once the model architecture is constructed and trained well, it is not complicated to transfer or even improve on other datasets and tasks. Such characteristics of the research subjects breed the conditions for developing classic structures such as U-Net (Ronneberger et al., 2015) and BERT (Devlin et al., 2018). Time series has the characteristics of multi-source complex, dynamic heterogeneity, missing anomaly, nonlinearity, uncertainty, etc., which makes it very difficult to achieve accurate prediction. The research challenges of time series forecasting mainly include two aspects. Challenge 1: Strong independence between different tasks and no standardized generic model exists. Researchers must conduct exclusive analyses and design specific models for a forecasting problem in data preprocessing, feature fusion, model selection, etc. Directly using a model architecture designed for a particular task is likely to perform poorly on other tasks. Forecasting the global infection number of COVID-19 is a powerful testament to this challenge [2]. Challenge 2: The

[2]During the COVID-19 outbreak, Kaggle, a well-known data mining competition platform, launched a competition to predict the daily COVID-19 infections (https://www.kaggle.com/c/covid19-global-forecasting-week-1). This competition comprises five rounds in total. Due to the one-sidedness and delay of epidemic data, as well as the complexity and changeability of social factors, the number of infections frequently emerges as it develops, which makes accurate forecasting more difficult. We found that the champion of each round of the competition is distinct, which indicates that it is challenging for one forecasting model to be invincible.

randomness and uncertainty of future values are enormous, and forecasting models with great generalization are very scarce. Time series are constantly evolving in dynamic situations. Its instantaneity, complexity, and uncertainty determine that any forecasting model cannot be omnipotent, and the forecasting result of a single model often holds the risk of high variance. The famous statistician George EP Box once said, "all models are wrong, but some are useful (Wikiquote, 2020)". Regarding this well-known saying within the scenario of time series forecasting, it implies that the ground truth of the future series is highly uncertain, and the predictive results of the model always retain errors.

1.2.2 Research Methodologies

Although the research of time series forecasting has not yet developed highly generic model architecture such as U-Net and BERT, researchers have also proposed numerous effective methods. Current methods can be divided into statistics-based and machine learning-based methods (Faloutsos et al., 2018). Statistics-based methods mainly include autoregressive integrated moving average (ARIMA) model, vector autoregressive (VAR), exponential smoothing, etc. (Hyndman and Athanasopoulos, 2014). These models have simple structures and great interpretability. They can only capture linear relationships among variables and hardly introduce external variables, and the model construction also requires strong domain knowledge and expert experience. Nevertheless, in the face of small data volume and superficial variable relationships, such statistics-based methods sometimes have acceptable performance (Hewamalage et al., 2019). Traditional machine learning-based methods include support vector machine, matrix factorization, Gaussian process, Bayesian network, artificial neural network, etc.

However, these shallow models' learning ability and generalization are limited when modeling complex nonlinear relationships among numerous variables. As a cutting-edge method in machine learning, deep learning has the outstanding capability of automatic feature extraction and has attracted remarkable attention from the research community of time series forecasting (Benidis et al., 2020; Faloutsos et al., 2018; Lim and Zohren, 2020). Since deep learning is essentially a deep neural network, researchers refer to such methods as *deep forecasting* or *neural forecasting*. A variety of deep learning models have been designed for different prediction tasks, mainly divided into convolutional neural networks, recurrent neural networks, and hybrid neural networks. Deep learning is prone to overfitting when modeling small data with simple variable relationships, sometimes hard to be satisfactory, but it has obvious advantages when modeling big data with high dimensions and complex variable relationships (Hewamalage et al., 2019). With the generation of time series big data, deep learning has become the mainstream approach to handling time series forecasting tasks. Some scholars have also begun exploring the possibility of combining traditional statistics-based methods with deep learning to improve forecasting accuracy. For example, Smyl has found that combining exponential smoothing and deep learning can reach more satisfactory performance (Smyl, 2020).

1.3 Research Questions and Objectives

This research aims to devise effective deep forecasting methods, address real-world forecasting problems, and promote the study of deep forecasting techniques. This research will answer research questions as below:

RESEARCH QUESTION 1 (RQ1) How to develop effective deep forecasting models in spatio-temporal forecasting scenarios?

RESEARCH QUESTION 2 (RQ2) How to quantify the predictive uncertainty with specific distribution assumptions for the deep forecasting model?

RESEARCH QUESTION 3 (RQ3) How to quantify the predictive uncertainty without distribution assumptions for the deep forecasting model?

RESEARCH QUESTION 4 (RQ4) How to design the loss function to improve forecasting accuracy and robustness?

To answer these research questions, this research aims to achieve the following objectives:

RESEARCH OBJECTIVE 1 (RO1) (in answer to RQ1) To develop a deep forecasting network based on noisy residual connection and a snapshot ensemble to reduce the training time and improve generalization.

This objective corresponds to research question 1. Aiming at the problems of the long training time of deep networks for point estimation and the weak generalization of a single model, a point estimation network based on noisy residual connection and snapshot ensemble forecasting methods is proposed. The model reasonably simplifies the input data and introduces Gaussian noise to the residual connection to achieve regularization so that the model can significantly reduce the training time while ensuring generalization accuracy. The method of forecasting using the snapshot ensemble for point estimation has also been explored. It can substantially improve the forecast accuracy without much difference from the training time of a single model. The experiment is evaluated based on two realistic traffic flow datasets, and the results confirm the effectiveness of the proposed methods.

RESEARCH OBJECTIVE 2 (RO2) (in answer to RQ2) To develop a deep uncertainty quantification network with Gaussian assumptions.

This objective corresponds to research question 2. Aiming at how to quantify uncertainty in forecasting, a deep uncertainty quantification method is proposed. The approach suggests a new likelihood loss function based on the Gaussian distribution as the training target of the encoder-decoder model, allowing the model to simultaneously predict the point estimation and prediction interval of multiple variables at multiple steps in the future. The research also reported the generalization improvement generated by training the deep forecasting model using the likelihood loss function. This research applies this method to real weather forecasting problems. By designing an effective multi-source information fusion mechanism, it can improve the forecast accuracy of various meteorological variables and estimate their prediction interval.

RESEARCH OBJECTIVE 3 (RO3) (in answer to RQ3) To develop a deep uncertainty quantification network without distribution assumptions.

This objective corresponds to research question 3. To further solve the problem that the true distribution of the target variables in practice cannot be foretold, a deep uncertainty quantification method without distribution assumption is proposed. The method constructs a novel loss function without distribution hypothesis, the so-called distribution-free loss function, as the objective function of training the encoder-decoder to capture data distribution adaptively in the training stage and then output point estimation and prediction interval in the forecasting stage. The effectiveness of the proposed method is verified on three public datasets.

RESEARCH OBJECTIVE 4 (RO4) (in answer to RQ4) To develop a deep quantile fusion network for high accuracy and great robustness.

This objective corresponds to research question 4. Aiming at the problems of low accuracy and weak robustness in prediction, a novel deep quantile fusion method for point estimation is developed. By designing a novel loss function, the method can transform the hidden layer into a quantile layer with semantic information and take quantiles as the input feature of the following layer to forecast the target variable, thereby constructing a deep forecasting model with high accuracy and excellent robustness. The experiments are evaluated on eight UCI regression datasets and one time series dataset, and the results validate the effectiveness of the proposed methods.

1.4 Research Significance

With the rapid popularization of sensing networks and storage capability, spatial and temporal data are growing more and more sufficient. These data resources lay a solid foundation for research in time series forecasting that has both theoretical and practical significance.

From the perspective of research, time series forecasting is a research hotspot in the field of artificial intelligence. In recent years, the number of submissions to the annual IEEE and ACM top conferences on deep forecasting has dramatically increased, which implies how much attention researchers are paying to developing deep learning models and improving time series forecasting. This thesis proposes several methods that utilize deep learning models to enhance the performance of time series forecasting. Specifically, this is the first research to introduce Gaussian noise to the residual network as regularization so that the deep model can achieve better generalization. This thesis is the first to present deep forecasting using

the snapshot ensemble can significantly improve the accuracy without consuming much training time. Moreover, this research provides various ways to improve the accuracy of forecasting and contributes to researchers with a better understanding of uncertainty and instability in forecasting problems. In particular, two types of deep uncertainty quantification methods are constructed, and researchers who have recently entered the field can gain deep insights from this research. Lastly, this thesis highlights model robustness as a research focus for forecasting models. This inspired the concept of deep quantile fusion to model the forecasted variable in a way that fuses original input features with multiple quantiles. This is an innovative way to achieve high accuracy and excellent robustness in deep forecasting models.

From the perspective of practice, time series widely exist in essential fields such as transportation, weather, finance, sales, and so on. Time series forecasting plays a vital role in the operation of modern society. Effective forecasting is conducive to reducing costs and helping managers take precautions. For example, product supply and demand forecasting can help optimize inventory management and personnel scheduling, thereby optimizing the overall resource allocation in the upstream and downstream of the supply chain; traffic flow forecasting can help analyze the traffic patterns of the traffic network in specific time periods, which is conducive to alleviating traffic congestion and improving traffic efficiency is an essential part of the intelligent transportation system; in addition, time series forecasting is very common in urban computing, weather forecasting, air quality prediction, financial markets, and many other fields.

1.5 Thesis Structure

The logical structure of this thesis (the chapters and the corresponding research questions) and the relationship between the chapters are shown in Figure 1.1. The main contents of each chapter are summarized as follows:

Chapter 2 presents a systematic literature review of research related to this study. In this chapter, we review related studies from three perspectives: deep forecasting, uncertainty quantification, and popular applications. We begin with a general introduction to deep forecasting, followed by a discussion of model architectures and loss functions. Next, we define uncertainty quantification and describe recent research developments in this area. Finally, we review popular applications of deep forecasting. Through this comprehensive review, we identify and summarize current research gaps.

Chapter 3 proposes a point estimation network based on noisy residual connections and a snapshot ensemble forecasting method to address the long training time and weak generalization of deep learning for point estimation. The proposed model simplifies the input data and introduces Gaussian noise to the residual connections to achieve regularization, reducing training time while maintaining generalization accuracy. The proposed ensemble method improves predictive accuracy without much additional training time. The experiment is evaluated using two realistic traffic flow datasets, and the results confirm the effectiveness of the proposed methods.

Chapter 4 proposes a deep uncertainty quantification method to quantify uncertainty in forecasting. The method proposes a new likelihood loss function based on the Gaussian distribution as the training target of the encoder-decoder model, allowing the model to simultaneously predict the point estimation and

prediction interval of multiple variables at multiple steps in the future. This research applies this method to real-world weather forecasting problems. By designing an effective multi-source information fusion mechanism, it can improve the forecast accuracy of various meteorological variables and estimate their prediction interval.

Chapter 5 presents a deep uncertainty quantification method without distribution assumption for addressing the problem that the true distribution of the target variables in practice cannot be foretold. The method constructs a novel loss function without a distribution hypothesis. In the training stage, the loss function can capture data distribution adaptively and then output point estimation and prediction interval simultaneously in the forecasting stage. The effectiveness of the proposed method is verified on three public datasets.

Chapter 6 develops a novel deep quantile fusion method for point estimation to handle the problems of low accuracy and weak robustness in prediction. By designing a novel loss function, the method can transform the hidden layer into a quantile layer with semantic information and take quantiles as the input feature of the following layer to forecast the target variable, thereby constructing a deep forecasting model with high accuracy and great robustness. The experiments are evaluated on eight UCI regression datasets and one time series dataset, and the results validate the effectiveness of the proposed methods.

Chapter 7 summarizes the contributions of this research and discusses research issues for further study.

It should be justified that each chapter of Chapter 3 to Chapter 6 in the thesis focuses on a specific objective, and the methods and datasets used in each chapter are carefully selected to achieve that objective best. Furthermore, the datasets used in each chapter are also chosen based on their reliability and validity. We carefully

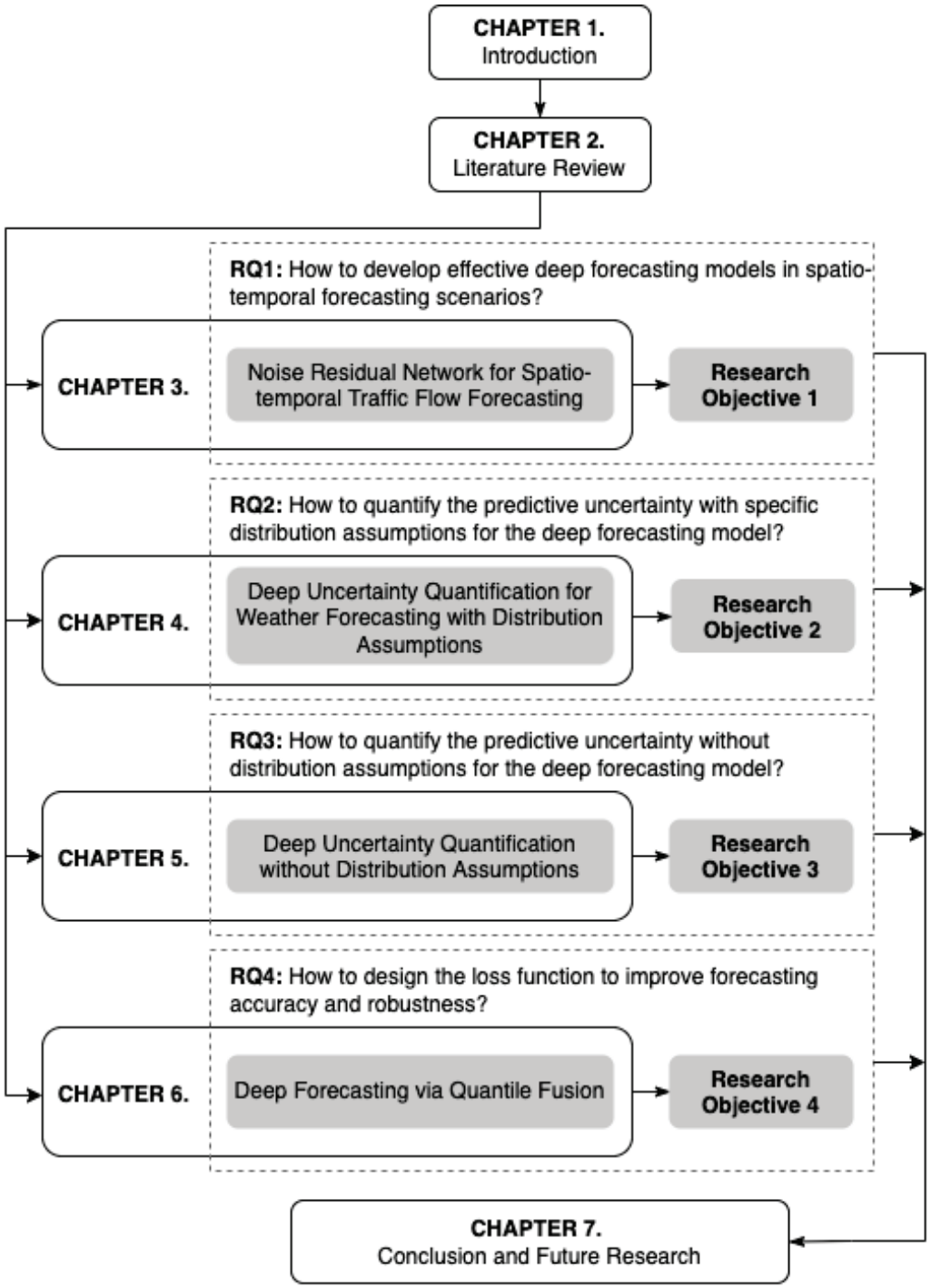


Figure 1.1 Thesis structure

evaluate the sources and methods used to collect the data and only use datasets that are deemed to be of high quality and relevance to the research question at hand. Overall, the use of different datasets in different chapters allows us to conduct a thorough and comprehensive analysis of the research questions and provides a strong foundation for the conclusions presented in the thesis.

1.6 Publications Related to This Thesis

Following is a list of the refereed international journal and conference papers during my PhD research that have been published:

Published:

1. **B. Wang**, Z. Yan, J. Lu, G. Zhang and T. Li. "Explore uncertainty in residual networks for crowds flow prediction," in *Proceedings of the International Joint Conference on Neural Networks* (*IJCNN*), pp:1-7, IEEE, 2018 (CORE Ranking B)

2. **B. Wang**, J. Lu, Z. Yan, H. Luo, T. Li, Y. Zheng and G. Zhang. "Deep uncertainty quantification: A machine learning approach for weather forecasting," in *Proceedings of the Twenty-fifth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (*KDD*), pp:2087-2095, ACM, 2019 (CORE Ranking A*)

3. **B. Wang**, T. Li, Z. Yan, G. Zhang and J. Lu. "DeepPIPE: A distribution-free uncertainty quantification approach for time series forecasting," *Neurocomputing*, 2020, vol. 397, pp. 11-19, 2020. (CORE Ranking B)

4. **B. Wang**, Jie Lu, T. Li, and G. Zhang. "A quantile fusion methodology for deep forecasting," *Neurocomputing*, 2022, vol. 483, pp. 286-298, 2022. (CORE Ranking B)

5. **B. Wang**, Z. Yan, J. Lu, G. Zhang and T. Li. "Deep multi-task learning for air quality prediction," in *Proceedings of International Conference on Neural Information Processing* (*ICONIP*), pp: 93-103, 2018 (CORE Ranking B)

6. **B. Wang**, Z. Yan, J. Lu, G. Zhang, and T. Li. "Road traffic flow prediction using deep transfer learning" in *Proceedings of International FLINS Conference*, pp: 331-338, 2018

Chapter 2

Literature Review

This thesis focuses on solving time series forecasting problems based on deep learning techniques. Since deep learning is essentially a deep neural network, many researchers refer to such methods as *deep forecasting* or *neural forecasting*. This chapter presents a literature review of relevant studies in this area. We review the related studies from three aspects: deep forecasting, uncertainty quantification, and popular applications. In Section 2.1, the most popular model techniques are introduced. This is followed by a formal definition of uncertainty quantification and a review of deep uncertainty quantification in Section 2.2. Finally, the selected real-world applications of deep forecasting are discussed in Section 2.3.

2.1 Deep forecasting

In recent years, ground-breaking achievements of deep learning in computer vision and natural language processing have aroused interest in applying deep learning techniques to time series forecasting, and such methods are termed *deep forecasting*. This section reviews the related studies from two perspectives, i.e., novel model architectures and model loss functions. There are many deep forecasting models based on CNN and RNN. However, in our experience, they are not all very effective and efficient. Subsection 2.1.1 selectively introduces the studies based on the most two popular and effective deep model architectures. Subsection 2.1.2 surveys the studies that designed novel loss functions to train the deep models.

2.1.1 Model architectures

Residual networks

He et al. first proposed the deep residual networks that reformulated the layers as learning residual functions with reference to the layer inputs instead of learning from scratch (He et al., 2016a). Essentially, residual networks can be thought of as a special case of highway networks (Zilly et al., 2017). Due to the non-linear activations still existing after the identity map, which can bring about gradient vanishing, He et al. improved the original residual networks and proposed the identity connection concept (He et al., 2016b). Researchers have recently published several papers augmenting the ResNet model with exciting improvements. Targ et al. proposes a architecture called ResNet in ResNet to generalize the ResNet (Targ et al., 2016). Liao and Poggio applies residual units into recurrent nets and

identified such models have enhanced performance (Liao and Poggio, 2016). Shah et al. combined exponential linear units, an alternative to rectified linear units, with ResNet to show improved performance, even without batch normalization (Shah et al., 2016). Zhang et al. proposed ST-ResNet, which is an improved version of DeepST (Zhang et al., 2017a). To get better precision, it utilized a residual learning module to make the model deeper and learn better.

Time series Transformers

As an emerging research subject in deep forecasting, time series Transformer has shown a great advantage for capturing long-range dependencies and periodicity, and thus is gradually attractive for time series forecasting. The vanilla Transformer was proposed by Vaswani et al. to address translation tasks (Vaswani et al., 2017). It is composed of an encoder-decoder architecture, and both encoder and decoder consist of multi-head self-attention and feed-forward networks without recurrence and convolution, hence improving computing efficiency. Positional encoding is adopted to encode the order information of input time series (Vaswani et al., 2017). Since Transformer has exhibited excellent modeling performance for sequential data, various modifications of Transformer have been designed to handle time series modeling (Wen et al., 2022). As a Transformer-based variant, Informer is proposed especially for the longer time series forecasting. It incorporated the learnable temporal embeddings and devised probabilistic attention, which dramatically reduces the computational complexity (Zhou et al., 2021). Autoformer employs a periodicity decomposition module with an Auto-Correlation mechanism as a novel attention mechanism, allowing it to yield state-of-the-art accuracy for long-term forecasting (Wu et al., 2021). FEDformer designs a frequency-enhanced decomposed Transformer to model global seasonality and adopts Fourier-

enhanced blocks with Wavelet-enhanced blocks to capture important structural information through frequency domain mapping for time series modeling (Zhou et al., 2022). TFT adopts recurrent layers for processing local information and interpretable self-attention layers for modeling long-term dependencies. It hence preserves useful components and suppresses unnecessary features (Lim et al., 2021). Aliformer proposes a knowledge-guided attention layer to take advantage of the future knowledge and introduces the future-emphasized training strategy to help the model focus more on the knowledge (Qi et al., 2021). Pyraformer devises a pyramidal attention module to model temporal dependencies of diverse ranges with linear time and memory complexity (Liu et al., 2021).

2.1.2 Model loss functions

Generic deep learning methodologies can be constructed from the following perspectives (Ian Goodfellow and Courville, 2016): 1) Basic architectures, such as Seq2Seq, Graph Neural Networks, etc. 2) Effective modules, such as attention mechanism, normalization layer, etc. 3) Optimization algorithms, such as Adam, RMSProp, etc. 4) Loss functions, such as MAE/MSE in regression and cross-entropy loss in classification. Among these aspects, designing novel architectures and modules becomes pervasive. Nevertheless, a recent keynote (Rudin, 2019) highlighted a philosophy that *it is not clear that extra and complex structure is always necessary or helpful. Models can often be made easier by adding interpretability constraints, which shrink the hypothesis space*. The less noticeable aspect is the usage of loss functions. A recent study of weather forecasting (Wang et al., 2019a) reported a novel phenomenon that training the vanilla Seq2Seq model

with Gaussian-based log-likelihood loss function instead of MSE can significantly improve the model generalization.

Quantile regression (QR) has been proposed by designing the quantile loss, a.k.a. pinball loss, to estimate conditional quantiles to model distribution information (Koenker and Bassett Jr, 1978). With the renaissance of deep learning, a combination of QR and deep neural networks has attracted considerable attention (Rodrigues and Pereira, 2020; Yan et al., 2018; Zhang et al., 2019). iQRNN (Zhang et al., 2019) incorporates quantile loss in deep neural networks to generate multiple quantile forecasts for load forecasting. DeepJMQR (Rodrigues and Pereira, 2020) was first proposed to train deep learning models by combining quantile loss and MSE together, which can play a regularization role in forecasting accuracy and reduce quantile crossing. In study (He et al., 2021), authors proposed a novel SVR model to improve QR for wind power probabilistic forecasting. The approach in (Hu et al., 2021) extended quantile regression for distributed massive datasets. With deep learning for forecasting attracting considerable attention, recent studies have begun to use deep quantile regression to predict the distribution and quantify uncertainty. Zhang et al. adopted deep learning techniques to improve traditional QRNN (Zhang et al., 2019). The improved model iQRNN incorporated noise layer, batch training, early stopping, and adaptive learning rate. It obtained excellent quantile forecasting and prediction interval and saved computational time and memories. LSTM has been combined with QR to learn the dynamic tail behaviors of financial asset returns (Yan et al., 2018). The authors utilized a parametric heavy-tailed quantile function to approximate the conditional distribution of financial return and formulated the learning of the parameters in a quantile regression fashion. Dabney et al. proposed QR to learn value distribution instead of the value function for reinforcement

learning (Dabney et al., 2018). They first constructed N adjustable locations for the approximation distribution and then deployed QR to minimize the Wasserstein metric to a target distribution. In study (Deng et al., 2020), a multiscale CNN was devised to extract features for forecasting quantiles. The log-cosh loss function was adopted to constrain the network in the training phase. Experiments showed that the model was efficient in quantifying the uncertainty of electricity load forecasting. All these studies demonstrate that the capability of deep forecasting models may not only be related to its complex architecture but also be related to how to utilize loss functions wisely.

2.2 Uncertainty quantification

Although deep forecasting has achieved great success, methods like (Yi et al., 2018; Zhang et al., 2018; Zhou and Matteson, 2015) only focus on the *single point forecasting*, a.k.a. *point estimation*, but ignore the *uncertainty quantification*, a.k.a. *prediction interval*. Deep learning can automatically extract effective features that can be exploited for high-quality uncertainty quantification. This section first introduces the basic concept of uncertainty quantification, then gives a systematic review of the related studies.

2.2.1 Concept of uncertainty quantification

For ease of understanding uncertainty in forecasting scenarios, let us first consider the equation (Abdar et al., 2021; Pearce et al., 2018): $\hat{\mathbf{Y}} = f(\mathbf{X}) + \varepsilon$, where statistically $f(\cdot)$ is the mean estimation (predictable point estimation) of the learned machine learning model, which is also called the epistemic uncertainty.

This uncertainty is caused by *model variance* and denoted as σ_m^2; For *data variance*, ε is the irreducible noise, which is also termed as the aleatoric uncertainty. The reason it exists is that there are unobtained explanatory variables or unavoidable random factors. Due to the difficulty of expressing ε with a deterministic equation, data variance is usually modeled by a Gaussian distribution with zero mean and a variance σ_d^2 (Central Limit Theorems). If σ^2 does not change, it is a homoskedastic problem; otherwise, it is regarded as heteroscedastic. Then the total variance $\sigma^2 = \sigma_d^2 + \sigma_m^2$. The learning process is usually implemented by maximum likelihood estimation, which will learn the estimated $\hat{\sigma}^2$.

Uncertainty quantification can provide more reference information for decision-making and has received increased attention from researchers in recent years (Khosravi et al., 2011a). However, most uncertainty quantification methods are based on shallow models and do not take advantage of deep learning.

2.2.2 Deep uncertainty quantification

Time series forecasting is a problem of dynamic modeling in nature, which usually takes on non-linear, uncertain and chaotic properties. Detecting chaotic behavior from a time series has been proposed (Kodba et al., 2005). Although it has been found that chaotic systems are ubiquitous, most of them do not have explicit dynamical equations and can be only modeled through the observed time series (Liu, 2010). In recent years, with deep learning reaching fever pitch in data science, various types of deep neural networks were introduced to model time series forecasting (Chen et al., 2018a; Deng et al., 2017; Qiu et al., 2017; Zhao et al., 2018; Ziat et al., 2017). Most of these studies focused on point estimation, and have devised effective network architectures such as sequence-to-sequence (Xiao et al.,

2018), attention mechanism (Yuan et al., 2018), embedding (Chen et al., 2018b), which can shed enlightenment on their combination with uncertainty quantification.

Deep learning can automatically extract desirable representations, which is very promising for high-quality uncertainty quantification. To this end, Bayesian deep learning (BDL), which learns a distribution over weights, is currently the most popular technique (Wang and Yeung, 2016). Nevertheless, BDL has a prominent drawback in that it requires significant modification, adopting variational inference (VI) instead of back-propagation (BP), to train deep models. Consequently, BDL is often more difficult to implement and computationally slower. An alternative solution is to incorporate uncertainty directly into the loss function and directly optimize neural networks by BP (Lakshminarayanan et al., 2017; Nix and Weigend, 1994; Pearce et al., 2018). This still suffers from certain limitations. 1) Regression is solved as a *mapping* problem rather than *curve fitting*, hence this method cannot naturally be applied to multi-step time series forecasting (Pearce et al., 2018; Roberts et al., 2013). 2) The output only considers a single dimension. If it is to be extended to multiple dimensions or for multi-step time series forecasting, the method must be based on effective and reasonable assumptions. 3) Only a shallow forward neural network is used for illustration, and the superior performance of deep learning is not explored.

Deep uncertainty quantification is constantly a hot research topic. A pioneering neural-network-based method is MVE, which utilizes the neural network to predict mean and variance to parameterize Gaussian distribution (Nix and Weigend, 1994). This method has been progressively combined with advanced network architectures such as echo-state network (Yao et al., 2017) and deep forward neural networks (Lakshminarayanan et al., 2017). However, MVE relies on a strong assumption that

the irreducible noise ε_{t+1} obeys Gaussian distribution and performs poorly when the assumption is too inconsistent with reality. To this end, two distribution-free neural network-based methods are especially proposed. Method LUBE constructed a non-convex loss function and hence used simulated annealing for optimization (Khosravi et al., 2010). Method DeepHQ is optimized based on gradient descent-based algorithms (Pearce et al., 2018). Nonetheless, both LUBE and DeepHQ can only implement prediction intervals without point estimation and conduct regression for cross-sectional data rather than time-series data, which cannot be directly applied to time series scenarios (Roberts et al., 2013). A Bayesian method was proposed for modeling time series (Johnson and Willsky, 2014). However, Bayesian methodology required Markov chain Monte Carlo (MCMC) or variational inference (VI) as the optimization algorithm, and when combined with deep learning as Bayesian deep learning (BDL) (Fortunato et al., 2017), it causes an efficient bottleneck of implementation. Non-Bayesian deep learning methods hence have been proposed to address the issue of BDL. A non-Bayesian study mathematically proved that Monte Carlo Dropout (MC-Dropout) could be operated as Bayesian approximation (Gal and Ghahramani, 2016) hence can be taken as a solution based on the back-propagation algorithm. The following research extended MC-Dropout for modeling time series based on recurrent neural networks (Zhu and Laptev, 2017). Another branch, i.e., non-Bayesian approach, is to design a novel loss function to depict uncertainty quantification when modeling time series. Method DeepMVE, inspired by MVE and sequence-to-sequence architecture, was proposed based on the assumption that the irreducible noise ε_{t+s} obeyed Gaussian distribution (Wang et al., 2019a). However, this method may overestimate the lower and upper bounds when the actual noise is non-Gaussian. DE-RNN (Yeo et al., 2018) was

proposed, which did not require the distribution assumption of ε_{t+s}, but DE-RNN is limited in that: 1) It needs discretization to transform regression as a classification problem, which asks users to denote the interval width at the expense of information loss; 2) It adopts the Monte Carlo procedure for multi-step forecasting, which is time-consuming; 3) It evaluates the aspect of PICP alone but ignores the quality of interval width.

2.3 Applications of deep forecasting

Time series forecasting plays an essential role in modern society. Many real-world problems are solved based on forecasting techniques. It ranges from energy forecasting (He et al., 2021; Taieb et al., 2017) and volume prediction (Diao et al., 2019; Feng et al., 2019; Guo et al., 2019; Lin et al., 2019; Pan et al., 2019; Wang et al., 2017b) to click-through rate prediction (Zhou et al., 2019), etc. Reasonable and practical forecasting is conducive to reducing costs and helping managers take precautions. Deep forecasting has been widely applied in urban computing, weather forecasting, air quality prediction, financial markets, and many other real-world fields (Torres et al., 2021). This section introduces the development of deep forecasting with regard to two most popular and typical application scenarios, i.e., flow prediction (Chen et al., 2021) and weather forecasting (Ren et al., 2021).

2.3.1 Flow prediction

Hoang et al. first proposed the citywide flow prediction. In particular, they divided the flows into three components and then utilized Gaussian Markov random fields and residual model to model these components. Inspired by the great success

of deep learning on computer vision, Zhang et al. first applied deep learning to citywide flow prediction and proposed DeepST (Zhang et al., 2016). This model adopted three DCNNs to capture closeness, period, and seasonal trend, respectively. Besides spatio-temporal data, it also fused different source data, such as weather and for a better generalization. Furthermore, Zhang et al. proposed ST-ResNet, which is an improved version of DeepST. To get better precision, it utilized a residual learning module to make the model learn deeper (Zhang et al., 2017a). With a similar philosophy to predicting citywide crowds flow, Ma et al. adopted DCNN on grid-based spatiotemporal data to make transportation network speed prediction (Ma et al., 2017). Wang et al. applies ST-ResNet to real-time crime forecasting(Wang et al., 2017a). Ke et al. propose FCL-Net to forecast of passenger demand under on-demand ride services (Ke et al., 2017). Xingjian et al. introduced ConvLSTM 2015 (Xingjian et al., 2015). Ghaderi et al. proposed Deep Learning-based Spatio-Temporal Forecasting using LSTM(Ghaderi et al., 2017). Wu and Tan proposed short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework (Wu and Tan, 2016).

2.3.2 Weather forecasting

Weather forecasting has been well studied for more than a century. Most contemporary weather forecasting relies on the use of NWP approaches to simulate weather systems using numerical methods (Marchuk, 2012; Richardson, 2007; Tolstykh and Frolov, 2005). Some researchers have addressed weather forecasting as a purely data-driven task using ARIMA (Chen and Lai, 2011), SVM (Sapankevych and Sankar, 2009), forward neural network (Voyant et al., 2012), etc. These shallow models explore only a few variables, which may not

capture the spatio-temporal dynamics of diverse meteorological variables. Deep learning has also shown promise in the field of weather prediction. The study in (Hernández et al., 2016) first adopted an auto-encoder to reduce and capture non-linear relationships between variables and then trained a multi-layer perceptron for prediction. In (Grover et al., 2015), a deep hybrid model was proposed to jointly predict the statistics of a set of weather-related variables. The study in (Xingjian et al., 2015) formulated precipitation nowcasting as a spatio-temporal sequence forecasting problem and proposed convolutional LSTM to handle it. However, these purely data-driven models are limited in that: 1) they all ignore important prior knowledge contained in NWP, which may not capture the spatio-temporal dynamics of diverse meteorological variables; 2) some need tedious feature engineering, such as extracting seasonal features as inputs and kernel selection, which seems contrary to the end-to-end philosophy of deep learning; 3) all lack the flexibility of uncertainty quantification.

In summary, this chapter presents a literature overview of the existing deep forecasting methods and their limitations. In Chapter 2.1, we explore the popular deep forecasting methods, such as residual networks, time series transformer and diverse loss functions, and discuss their limitations in terms of accuracy. In Chapter 2.2, we delve into the concept of uncertainty quantification and introduce the development of deep uncertainty quantification and highlight their limitations in terms of flexibility. In Chapter 2.3, we discuss the current popular applications in time series forecasting and identify gaps in the existing methods that need to be addressed. The overall aim of this thesis is to contribute to the field of time series forecasting by developing novel deep forecasting methods that overcome the limitations of existing approaches. This will be achieved by exploring the potential

of the deep ensemble that combines the strengths of the powerful single deep model and snapshot ensemble and by developing novel loss functions that can handle uncertainty quantification and provide insights into the underlying distribution information of the forecasted variables. This thesis thus addresses the identified gaps in the existing methods and provides more accurate and flexible approaches to deep forecasting.

Chapter 3

Noise Residual Network for Spatio-temporal Traffic Flow Forecasting

Traffic flow forecasting is a typical time series forecasting problem, more precisely, a spatio-temporal forecasting problem. It is of great significance to public safety and traffic management and hence a core component in building intelligent transportation systems. Traffic flow data evolves simultaneously in two dimensions of time and space, which has high requirements on the ability of the forecasting model. Deep learning has the ability of capturing sequential spatio-temporal features and is naturally a considerable model technique. In this chapter, a deep point forecasting model is developed to handle the task of city-wide traffic flow prediction.

3.1 Introduction

Crowds flow is one type of traffic flow and predicting it is of great importance to the public safety and traffic management. For examples, during a morning rush hour in September 2017, a heavy downpour happened on a footbridge in Mumbai's Elphinstone station, a catastrophic stampede occurred and caused at least 22 people killed and dozens more injured. In Chinese 2015 New Year's Evening celebrations, a huge of people influxed into Shanghai Bund in a short time, giving rise to a disastrous stampede that killed 36 people. Effective preemptive countermeasures are highly desired to avoid or alleviate this tragedy by employing emergency measures in advance. For example, if heavy crowds in the next hour are predicted, then traffic regulators could immediately deploy police strength to evacuate people and release early warning. These preemptive countermeasures will only work when the traffic management authorities receive reliable predictions for future crowds flow. Previous research on crowds flow prediction mainly focused on the predictions of each road segment (Moretti et al., 2015) or individual movement (Ye et al., 2009). While these methods provide a detailed view of one segment, they are not required from the perspective of public safety and traffic congestion on large scale. In contrast, predicting crowds flow in a citywide rather than on a single road segment is significantly important to traffic control and city security. However, large-scale crowds flow prediction calls models for more powerful abilities, such as the ability to deal with highly spatial correlation among different areas incurred by the traffic topology, and the capability to capture temporal dependence and periodic characteristics.

This prediction task is intrinsically categorized as a high dimensional regression problem for a dynamical system and this study is primarily motivated by (Zhang

et al., 2017a) (Zhang et al., 2017a), they invented ST-ResNet to predict crowds flow, which is benefited from the residual learning (He et al., 2016a). The core insight of it is forcasting the future state by multi-source data fusion based on deep residual networks. However, due to ST-ResNet too deep and much inputs, it suffers from serious time consumption for training and this is not a acceptable property for online learning and ensemble learning. On the other hand, considering a dynamical system carries some degree of uncertainty, it may be due to an inherently stochastic process, a deterministic process which is partially from observation, or it may be due to the complexity of the process being greater than the capacity of the deep learning model. One reasonable way of improving model robustness is to inject uncertainty components, which can be made to account for aspects of the target that are not explainable from the observed input. Previous works has proved that noise injection plays an important constructive role in EM algorithm (Gulcehre et al., 2016) and activation functions of deep learning (Osoba et al., 2013).

Inspired by those, we for the first time explore noise injection for the residual networks by bringing in a Gaussian noise instead of the identity map within the residual learning architecture and design a compact architecture so as to be trained fast. It saves much training time and allows for predicting crowds flow with competitive accuracy. Meanwhile, our experiments show evidence that our method performs better than Dropout technique on this regression task. Following are the major contributions of this thesis.

1. We firstly propose to introduce noise components into residual networks instead of identity map for performance enhancement;

2. With the proposed method, we design a compact architecture for crowds flow prediction. Compared with ST-ResNet, Noise-ResNet reaches considerable

results and particularly attains the state-of-arts results on one benchmark dataset.

3. Save training time tremendously. Benefit from the light model. On one dataset, Noise-ResNet takes less nearly 90% training time for one epoch;

4. By comparing the proposed noise injection with the popular regularization method Dropout, we validate that our method can perform much better in this crowds flow prediction task. From another perspective, this can be regarded as a new regularization method for residual networks.

Although we only do experiments for crowds flow prediction, our methods also hold promise for other regression tasks, especially with apparent period and instantaneous variation such as traffic flow prediction and freight amount forcasting, etc.

The rest of this chapter is organized as follows: we introduce our work by some backgrounds in Section II. In Section III, we explain our models in more details. Then, we experimentally evaluate our proposed prediction models and compare them with other baselines in section IV. Finally, related works and conclusions of future study directions are given in Section V and VI, respectively.

3.2 Problem Statement

Before predicting the spatio-temporal flow in a specific region, it needs to divide the area into multiple grids and collect and count the flow data in each grid. The definition of spatio-temporal grids and traffic flow are described as follows:

Definition 1 - Spatio-temporal grids *We seperate a city into an $I \times J$ grid map based on the longitude and latitude where a grid stands for a region, as displayed in Figure 3.1.*

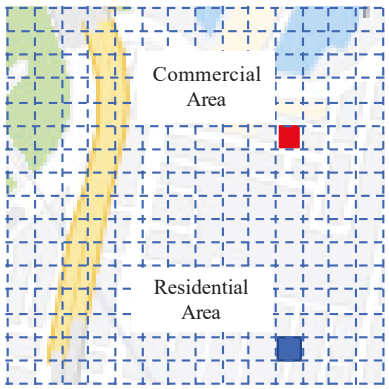


Figure 3.1 Spatio-temporal grids.

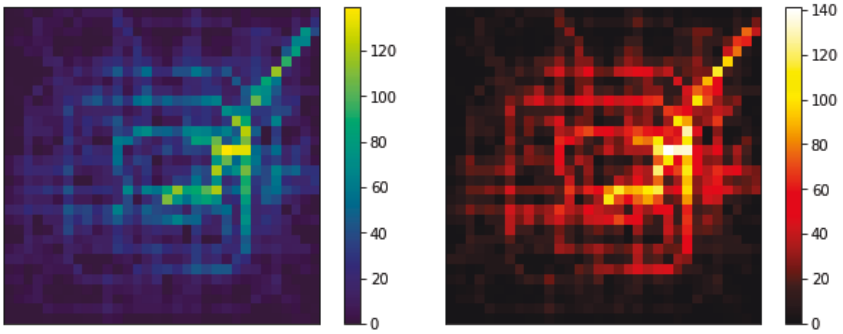
Definition 2-Inflow/Outflow *Let $\mathbb{P}$ be a collection of trajectories at the t^{th} time interval. For a grid (i, j) that locates at the i^{th} row and the j^{th} column, the inflow and outflow of the crowds at the time interval t are defined respectively as:*

$$x_t^{in,i,j} = \sum_{Tr \in \mathbb{P}} |\{k > 1 | p_{k-1} \notin (i,j) \wedge p_k \in (i,j)\}|$$

$$x_t^{out,i,j} = \sum_{Tr \in \mathbb{P}} |\{k \geq 1 | p_k \in (i,j) \wedge p_{k+1} \notin (i,j)\}|$$

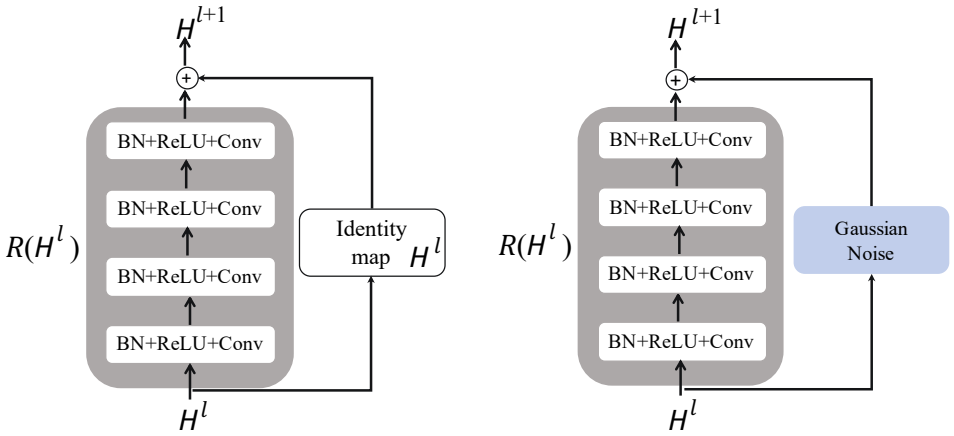
where $Tr : p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_{|Tr|}$ is a trajectory in $\mathbb{P}$, and p_k is the geospatial coordinate point; $p_k \in (i, j)$ means the point p_k locates within $grid(i, j)$; $|\cdot|$ denotes the cardinality of a set.

At the t^{th} time interval, inflow and outflow in all $I \times J$ regions can be represented as a tensor $X_t \in \mathbb{R}^{2 \times I \times J}$ where $(X_t)_{0,i,j} = x_t^{in,i,j}, (X_t)_{1,i,j} = x_t^{out,i,j}$. The flow matrix is shown in Figure 3.2.



(a) Heat map of outflow (b) Heat map of inflow

Figure 3.2 Spatio-temporal flow matrix



(a) (b)

Figure 3.3 (a) shows a residual unit consisting of 4 stacked BN+ReLU+Conv in ST-ResNet. (b) displays the residual unit with Gaussian regularization in Noise-ResNet.

Based on above, the citywide crowds flow at any time could be treated as a 2-channel image $X \in \mathbb{R}^{2\times I\times J}$.

Target *Given the historical crowds flow* $\{X_t|t=0,\ldots,n-1\}$, *forecast* X_n.

3.3 Proposed Methodology

3.3.1 Preliminary

Residual Learning With the increase of network depth, gradient vanishing seriously hindered the model training. The emergence of residual learning enables deeper neural networks can be trained successfully. The model that Microsoft used in ImageNet 2015 (Russakovsky et al., 2015) has 152 layers, even 1001 layers, which lead to state-of-the-art performance in both visual and non-visual tasks. The residual unit with an identity mapping is defined as (He et al., 2016a):

$$H^{l+1} = \sigma[H^l + \mathcal{R}(H^l)] \tag{3.1}$$

where H^l and H^{l+1} are the input and output of the l^{th} residual unit, respectively; $\mathcal{R}$ is a function for learning residuals, e.g., a residual unit consisting of 4 stacked BN+ReLU+Conv in ST-ResNet (Zhang et al., 2017a). The core insight of the residual learning is to learn from starting at the identity map function and gradually evolve to be more complex under the help of the additive residual term $\mathcal{R}$ with respect to H^l. σ is the activation function such as ReLU (Li and Yuan, 2017).

3.3.2 Model framework

The goal of Noise-ResNet is to assure high performance while reducing training time (Wang et al., 2018a). With this philosophy, Noise-ResNet is designed by considering three aspects: 1) Take as less inputs as possible. Different from ST-ResNet, we regardless period data and only takes as input closeness and trend data, which are considered as the two most important factors to capture present

and periodic property. 2) Only use one residual unit to reduce time consume. 3) To achieve better generalization, a key insight is that we introduce extra Gaussian noise instead of identity map into the residual learning.

The formulation of residual unit in Noise-ResNet is given as below:

$$H^{l+1} = \sigma[(H^l + G) + \mathcal{R}(H^l)] \tag{3.2}$$

where H^l, H^{l+1}, l^{th}, $\mathcal{R}$ and σ has the same meanings in formula (1); G represents the Gaussian noise.

Figure 3.4 presents the entire architecture of Noise-ResNet. As illustrated from bottom to top in Figure 3.4, we first turn inflow and outflow throughout a city at each time interval into a 2-channel image-like matrix respectively, using the approach introduced in Definitions 1 and 2. The inputs are comprised of two components including length-1 closeness data (X_{t-1}) and length-3 trend data ($X_{t-3w}, X_{t-2w}, X_{t-w}$). And then we concatenate all 2-channel flows into multiple channels as inputs of a convolutional layer (Conv1). Next only one noise residual unit followed by a ReLU activation, convolutional layer (Conv2), Tanh activation and MSE loss criterion to implement end-to-end learning.

Algorithm 3.1 outlines the Noise-ResNet training process. We first construct the training instances from the original sequence data (lines 1-6). Then, Noise-ResNet is trained via back-propagation and Adam optimization. Such an architecture could learn effectively and fast. which will be shown in experimental results.

3.3.3 Forecasting using snapshot ensemble

Snapshot Ensemble for Image Classification

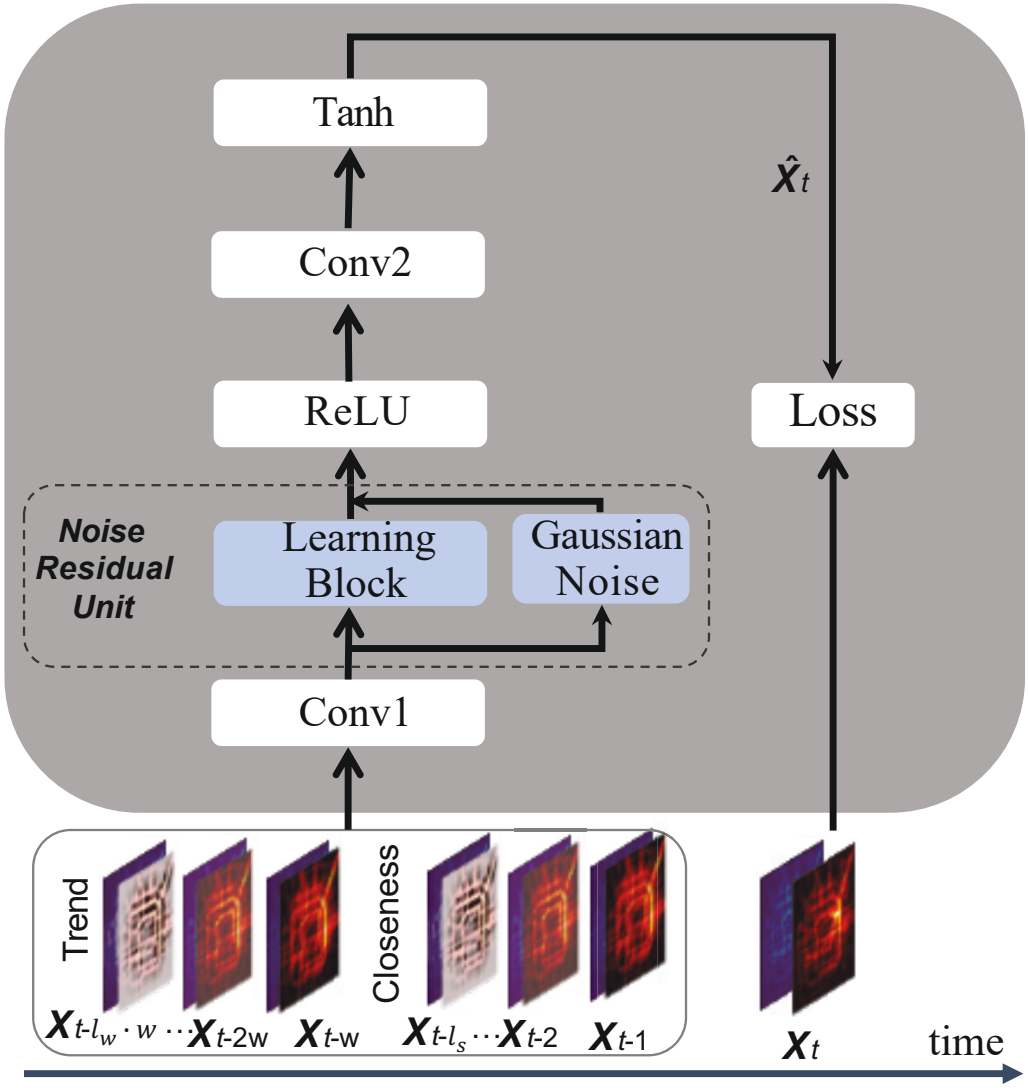


Figure 3.4 Model architecture of Noise-ResNet.

Snapshot Ensemble is first proposed as an efficient and effective deep ensemble technique for solving image classification problems (Huang et al., 2017). The snapshot ensemble prediction method FUSE proposed in this study is inspired by the original Snapshot Ensemble and improves Snapshot Ensemble for addressing prediction problems. In the process of training a single model, the FUSE method

Algorithm 3.1: Noise-ResNet Training

Input : Historical data: $X_0, ..., X_{t-1}$
length of *closeness*, *trend* sequences: l_h, l_w;
trend span: w.
Output : Trained Noise-ResNet

// Format training samples
$\mathcal{D} \leftarrow \emptyset$
while *all available time interval t*$(1 \leq t \leq t-1)$ **do**
  $S_h = [X_{t-l_h}, X_{t-(l_h-1)}, ..., X_{t-1}]$ $S_w = [X_{t-l_w \cdot w}, X_{t-(l_w-1)\cdot w}, ..., X_{t-w}]$
  // X_t *is the target at time t*
  put an training sample (S_h, S_w, X_t) into $\mathcal{D}$
end
// Train
Initialize all trainable parameters W in Noise-ResNet
repeat
  randomly select a batch of samples $\mathcal{B}$ from $\mathcal{D}$
  calculate W by minimizing the objective (3) with $\mathcal{B}$
until *stopping criteria is met*;

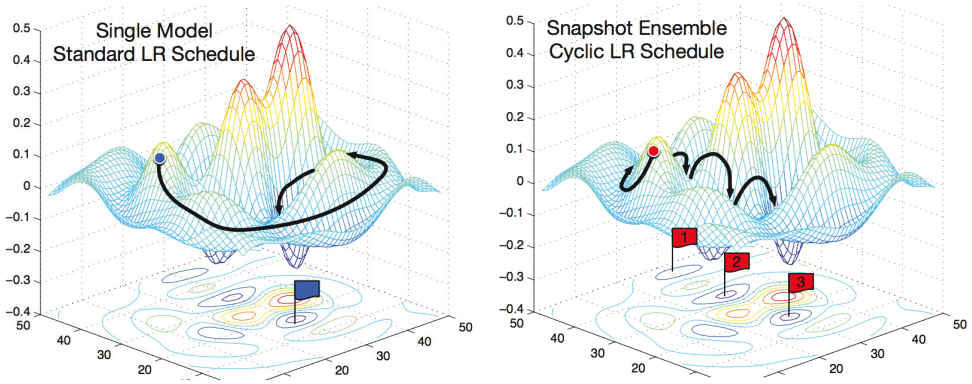
uses the cosine fire reduction method (formula 3.3) to dynamically adjust the learning rate:

$$\alpha_t = \frac{\alpha_0}{2}(cos(\frac{\pi \cdot mod(t-1, T/M)}{T/M}) + 1) \tag{3.3}$$

Specifically, if T represents the total number of iterations in the training phase [1], t represents the current number of iteration rounds, M represents the number of snapshots to be saved, the method is based on a large initial learning rate α_0 and gradually decays during training, so that the model quickly converges to a local optimal values, and then save a snapshot of the parameters of the model. Then the cosine annealing warm restarts adjusts the learning rate back to α_0 and decays again during the training process. This process is repeated for M times, and eventually M snapshots of the deep model are obtained. Figure 3.5b visualizes

[1]The model is trained once on all training sets, which is called 1 round, that is, 1 epoch.

this process. During the whole process, the model will go through multiple local optimal solutions (shown in red flags) and save the model weights at this time. In the integrated prediction stage, total M predictive results of the snapshot models are averaged as the final prediction result. In contrast, 3.5a shows the process of linear decay of the learning rate, and the model generally only converges to a local optimal solution (shown by the blue flag).



(a) Standard linear decay for the learning rate schedule

(b) Cosine annealing warm restart for the cyclic learning rate schedule

Figure 3.5 Comparison between two learning rate schedules (Huang et al., 2017)

Snapshot Ensemble for Time-series Forecasting Inspired by Snapshot Ensemble, the snapshot ensemble prediction method FUSE proposed in this study improves Snapshot Ensemble and applies it to prediction problems. FUSE contains the following two key points:

1) The configuration method of reasonable parameter values is explored. During the experiment, this study found that: a) The initial learning rate α_0 should be set moderately, and if α_0 is too small, it is not conducive to speeding up the training process in the early stage, nor is it conducive to escaping from the local optimum after each model snapshot is saved If the value of α_0 is too high, it will cause oscillations in training, which is not conducive to the convergence of the

model. Generally, α_0 is set to 0.001 to 0.01; b) The value of M should be set moderate, and the value of M is too small to give full play to the integration. The advantage of prediction. If the value of M is too large, the marginal effect of prediction performance will decrease due to too many deep models participating in the integration, and more storage space will be occupied. The value of M can generally be set to 5 to 10. c) The value of T should be set moderate. If T is set far small, each snapshot model will not be trained enough. If T is set very large, each snapshot model might be over-fitted, which can increase training time and also decrease generalization. The setup of value T can refer to the following principle. The operator first trains a single model and observe the loss variation on the validation set to determine the training epochs (i.e., the epochs in a snapshot period, abbreviated as E) that can achieve a relatively smaller validation loss. Accordingly, T is set as $E \times M$.

2) The Adam optimizer (Kingma and Ba, 2014) is used to replace the Stochastic Gradient Descent (SGD) optimizer in Snapshot Ensemble. Compared with SGD, Adam has many advantages, such as high computational efficiency, low memory usage, automatic adjustment of learning rate, etc. Therefore, Adam is the default optimizer with better performance. The experiment found that using the SGD optimizer to train the point estimation network, not only the training convergence speed is very slow, but also the network generalization performance is poor after training.

Figure 3.6 takes $\alpha_0 = 1 \times 10^{-3}, E = 30, M = 10, T = 300$ as an example to show the learning rate schedule of the FUSE in a snapshot period and displays the decaying variation for E (i.e., the value of T/M in the Formula 3.3) times. In a snapshot period, for each epoch, the latest α_t value will be calculated according to

the Formula 3.3, and the learning rate of Adam optimizer is updated accordingly. After decaying for E times, a snapshot model will be saved.

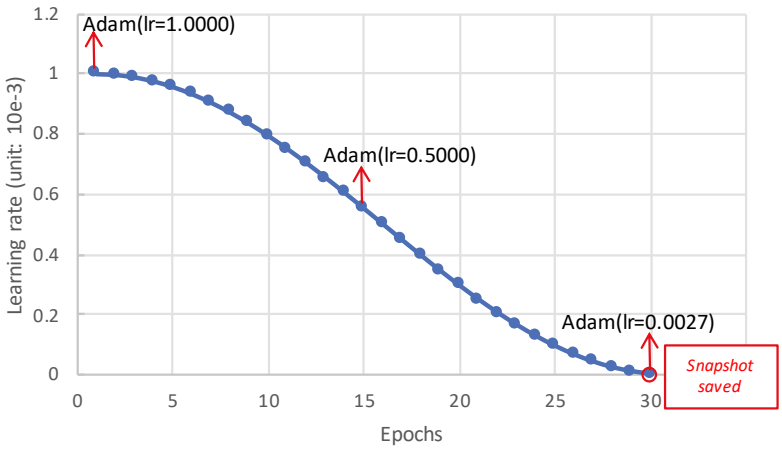


Figure 3.6 Illustration of dynamic schedule of the learning rate E times in one snapshot period

3.4 Experiments and Analysis

3.4.1 Datasets and preprocessing

We firstly use max-min normalization to preprocess data and then implemented experiments on two datasets as shown in Table 3.1.

Dataset BikeNYC is taken from the NYC Bike system in New York City in 2014 , from Apr. 1st to Sept. 30th. These trip data includes: trip duration, starting and ending station IDs, and start and end times. Among the data, the last 10 days are chosen as testing data, and the others as training data.

Dataset TaxiBJ includes the taxicab flow data and meteorology data in city Beijing, but our model only take as input the traffic flow data. It ranges from four time intervals: 1st Jul. 2013-30th Otc. 2013, 1st Mar. 2014-30th Jun. 2014, 1st Mar.

Table 3.1 Datasets TaxiBJ and BikeNYC

Dataset	TaxiBJ	BikeNYC
Data type	Taxi GPS	Bike rent
City	Beijing	New York
Time Span	7/1/2013-10/30/2013 3/1/2014-6/30/2014 3/1/2015-6/30/2015 11/1/2015-4/10/2016	4/1/2014- 9/30/2014
Grid size	32x32	16x8
Time interval	30 minutes	60 minutes
Available time interval	22,459	4392

2015-30th Jun. 2015, 1st Nov. 2015-10th Apr. 2016. According to the **Definition 2**, we obtain the input and output traffic flow. We choose data from the last four weeks as the testing data, and all data before that as training data.

3.4.2 Experimental settings and baselines

Environment settings

This subsection introduces the software and hardware environment and parameter settings related to the experiment. The operating system is Red Hat Linux workstation 7.5; GPU is Nvidia Quadro P4000, CUDA 10.0 and cuDNN 7.4; The coding environment is Python 3.5 and deep learning framework is Tensorflow 1.3.0. l_h and l_w are two hyperparamers in the network Noise-ResNet, which are the length of hour and week, respectively. We set l_h as 1 and l_w as 3. Learning rate is set as 0.0001. Dataset BikeNYC is relatively small, and to avoid overfitting when training the network Noise-ResNet on this dataset, we uphold the strong regularization and hence set the variance σ^2 of the Guassian noise as 0.5. Dataset TaxiBJ is relatively large, so we maintain the weak regularization and set the

variance σ^2 of the Guassian noise as 0.02 when training the network Noise-ResNet on this dataset. The convolution layer Conv1 and all residual units use 64 filters of size 3×3, and Conv2 is the convolution with 2 filters of size 3×3. Both two networks trained on BikeNYC and TaxiBJ only maintain one residual unit.

The batch size is 64. We further select 90% of the training data for training each model, and the remaining 10% is chosen as the validation set, which is used to early stop our training algorithm for each model based on the best validation score. Then we continue to train the model on the full training data for a fixed number of epochs (e.g., 30 epochs).

Baselines

Experimental baselines are as follows:

- **ARIMA** Auto-Regressive Integrated Moving Average (ARIMA) is a well-known method for modeling and predicting future values in a time series.
- **SARIMA** Seasonal Auto-Regressive Integrated Moving Average (SARIMA) is an extension variant based on ARIMA for modeling the cycle in a time series.
- **VAR** Vector Auto-Regressive (VAR) is an extension method based on ARIMA for capturing the correlation among multiple time series.
- **DeepST** DeepST is a spatio-temporal flow prediction model based on the fully connected deep neural network, which has once obtained the state-of-the-art performance on the city-wide flow prediction task (Zhang et al., 2016).
- **ST-ResNet** ST-ResNet divides the traffic flow data from the time dimension into three parts: close flow, trend flow and periodic flow. Each part of the

traffic data uses its own dedicated CNN subnet for feature extraction. In addition, there is an exclusive subnet for processing the external information, such as weather, time of day, etc. The four network branches are finally weighted by a learnable weight fusion layer to predict the future traffic value. For dataset BikeNYC, the method contains 4 residual units. For dataset TaxiBJ, the method contains up to 12 residual units (Zhang et al., 2017a).

- **Noise-ResNet** Since dataset BikeNYC is relatively small, the residual unit in Noise-ResNet stacks two combinations of BN+ReLU+Conv. In the experiment, the variance of Gaussian noise α^2 is set as 0.5, that is, $G \sim (0, 0.5)$. Dataset TaxiBJ is relatively large, which has higher requirements on the fitting ability of the model. Therefore, the residual unit in Noise-ResNet is stacked with 4 combinations of BN+ReLU+Conv to ensure that the model has sufficient learning capacity, and meanwhile the training speed is also considerable. Moreover, due to the large volume of dataset TaxiBJ, the variance α^2 of Gaussian noise G should not be set too large, otherwise the learning process will be destructed. In the experiment, α^2 is set as 0.02 to perform a light regularization, that is, $G \sim (0, 0.02)$. Additionally, different from ST-ResNet, Noise-ResNet only takes as input the traffic flow data without the external information.

- **Drop-ResNet** This baseline model is similar to Noise-ResNet, but the residual connection adopts Dropout to replace the noise perturbation regularization. The Dropout layer will randomly set some hidden layer features as zero, and the input features that are not set to zero will be set to 1/(1-dropout_rate). The ratio is scaled up so that the sum of all inputs remains the same. The baseline is taken as a control trial to compare the effect of Dropout and noise

perturbation regularization. The Dropout rate is set as 0.5 in dataset BikeNYC and 0.02 in dataset TaxiBJ.

- **Normal-ResNet** is similar to Drop-ResNet but without Dropout layer. This method aims to validate the effectiveness of Gaussian regularization.

- **FUSE** This method is an ensemble prediction based on snapshot ensemble. It saves multiple Noise-ResNet snapshot models during the training process. Corresponding to Equations 3.3, the method FUSE sets T as 300, M as 10 and α_0 as 0.001 in the experiment. In the prediction stage, the arithmetic mean of the each prediction result of M Noise-ResNet snapshot models is calculated as the final forecasted value of the ensemble method. Note that the method FUSE will stop training after the training process reaches T rounds, so the early stopping mechanism does not work.

3.4.3 Experimental evaluations and analysis

Evaluation metric

We measure all methods by *Root Mean Square Error* (RMSE) where $\hat{x}_i$ and x_i are the ith predicted value and ground thuth, respectively; z is the number of all predicted values.

$$RMSE = \sqrt{\frac{1}{z}\sum_{i}(x_i - \hat{x}_i)^2} \tag{3.4}$$

Note that during the training and testing stage, inputs and outputs have been preprocessed by min-max normalizaiton. In the experimental evaluations, the RMSE is rescaled to the realistic scale.

Experimental results of the single model

Table 3.2 displays the performances of different methods on dataset BikeNYC [2]. The performances of ARIMA, SARIMA, VAR, DeepST, and ST-ResNet all refer to the experimental results in the previous study (Zhang et al., 2017a). Due to different weight initialization, the trained deep model may have different prediction performance. In the experiment, Noise-ResNet, Drop-ResNet and Normal-ResNet were trained ten times with different initialization. The mean and standard deviation of each method were calculated based on the ten experimental results. Noise-ResNet achieved the best results with the mean RMSE of 6.10 and the variance of 0.06. By comparing Normal-ResNet and Noise-ResNet, it can be seen that the noise residual connection performs better than the identity mapping residual connection; by comparing Drop-ResNet and Noise-ResNet, it can be concluded that noise perturbation regularization performs better than Dropout regularization. At the same time, Noise-ResNet not only outperforms ST-ResNet which has more complicated structure in the predictive accuracy, but also reduces the training time by more than half compared to ST-ResNet.

Table 3.3 shows the experimental results on dataset TaxiBJ. Through the analysis of the results, the two main conclusions are: 1) ST-ResNet has better test performance than Noise-ResNet, mainly because the dataset TaxiBJ is larger, and the model ST-ResNet contains 12 residuals block, which endows its better fitting capacity on the big dataset, but it also consumes much training time. We can see that it takes nearly 7.3 hours in the training stage; 2)Noise-ResNet has a considerable generalization accuracy, even though not optimal, each training round has reduced nearly 90% training time compared with ST-ResNet, and it even takes 0.9 hours for a single model to be trained. Considering that the dataset TaxiBJ is

[2]More referred baselines could be found in the previous study (Zhang et al., 2017a).

Table 3.2 Comparison among different baselines on BikeNYC

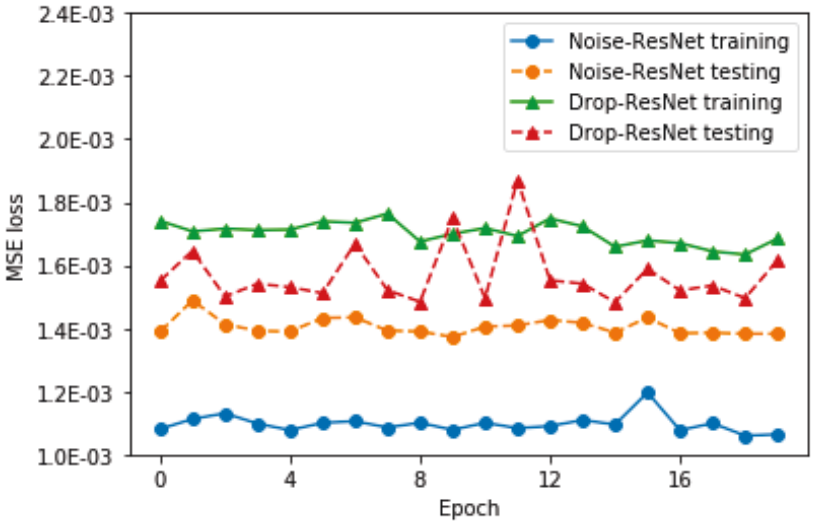
Model	RMSE	Std.	Time (s) / Epoch	Total Time (s)
ARIMA	10.07	-	-	
SARIMA	10.56	-	-	
VAR	9.92	-	-	
DeepST	7.43	-	-	-
ST-ResNet	6.33	-	3	432
Noise-ResNet	**6.10**	**0.06**	1	206
Drop-ResNet	6.52	0.19	-	-
Normal-ResNet	6.38	0.32	-	-
FUSE	5.51	-	1	300

Table 3.3 Comparison among different baselines on TaxiBJ

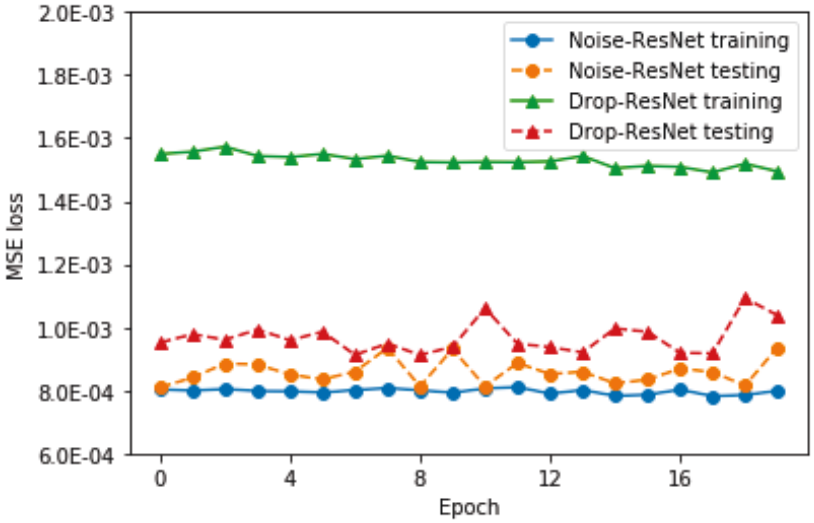
Model	RMSE	Std.	Time / Epoch	Total Time
ARIMA	22.78	-	-	
SARIMA	26.88	-	-	
VAR	22.88	-	-	
DeepST	18.18	-	-	-
ST-ResNet	16.69	-	$\sim$223s	$\sim$7.3h
Noise-ResNet	17.91	-	1	$\sim$0.9h
Drop-ResNet	19.29	-	-	-
Normal-ResNet	17.98	-	-	-
FUSE	16.63	-	$\sim$21s	$\sim$1.5h

too large and the training time is too long, we did not repeat many times to calculate the mean and standard deviation on this dataset. We would like to emphasize that Noise-ResNet has the advantage of greatly reducing the training time without losing much prediction accuracy on the large dataset.

To further explore the predictive performance of noise perturbation regularization and classic Dropout regularization, Figure3.7 displays the training and test loss curves for the last 20 epochs during the training stage. It can be seen that the training loss of Noise-ResNet is always lower than the test loss, while the training loss of Drop-ResNet is higher than the test loss. This study concluded that as a strong



(a) Loss variation on dataset BikeNYC for the last 20 epochs



(b) Loss variation on dataset TaxiBJ for the last 20 epochs

Figure 3.7 Loss changes of the last 20 epochs during training and testing

regularization, Dropout can cause the model underfitting, while noise perturbation will not corrupt the input features too much and bring in great generalization. We hence believed that noise perturbation is a more suitable regularization for addressing deep forecasting problems.

Performance of ensemble forecasting

The superior performance of the original Snapshot Ensemble has been experimentally verified in image classification tasks (Huang et al., 2017). This study improves Snapshot Ensemble and applies it to the deep forecasting problems, and proposes the method **F**orecasting **U**sing **S**napshot **E**nsemble (FUSE) for the tasks of time series forecasting. The experimental results demonstrated that FUSE has great performance. The experimental analysis is as follows.

As shown in Table 3.2, on the dataset BikeNYC, the RMSE of FUSE is 5.51, which is significantly exceptional than the single Noise-ResNet method. Even though the training time is about 100 seconds longer, and the performance improvement is appreciable. On the dataset TaxiBJ, the RMSE of FUSE is 16.63, which also significantly outperforms than the Noise-ResNet method and surpasses the ST-ResNet method. It is worth mentioning that ST-ResNet uses multi-source data as input, including close, trend, periodic traffic data, and external data such as weather and date information, and then introduces a fusion layer to achieve its state-of-the-art accuracy. The Noise-ResNet was trained using only short-term and trend traffic data. FUSE method based on the multiple trained Noise-ResNet models can obtain the state-of-the-art performance that exceeds ST-ResNet. In terms of training time, FUSE only takes about 1.5 hours to train $M = 300$ epochs, while ST-ResNet takes up to 7.3 hours.

In view of the superior generalization of FUSE, this study advocates that the FUSE method should be implemented as a standardized step when building deep forecasting models, which can help improve prediction accuracy significantly.

Sensitive analysis

This study takes the BikeNYC dataset as an example to conduct sensitivity analysis on the parameter σ^2 of Noise-ResNet. As shown in Figure 3.8, the

variance σ^2 of Gaussian noise is increased from 0 to 0.9 at 0.1 intervals to observe the prediction performance of Noise-ResNet after training. It can be seen that the RMSE of Noise-ResNet shows a trend of first decreasing and then increasing. When σ^2 equals to zero, Noise-ResNet is equivalent to Normal-ResNet, and the RMSE is 6.33; when σ^2 gradually increases, it is equivalent to reducing the structural risk of the model by enhancing regularization, but at the same time it will weaken the fitting ability of the model, that is, increase the empirical risk of the model. For example, when σ^2 equals to 0.9, Noise-ResNet has the largest generalization error, and the RMSE is 6.42. When σ^2 is properly selected, the model will achieve a balance between empirical risk and structural risk. For example, when $\sigma^2 = 0.2$, Noise-ResNet has the lowest generalization error and the RMSE is 6.20.

Similarly, the experiment also conducts sensitivity analysis on the dropout rate of Drop-ResNet. In the experiment, the dropout rate was increased from 0 to 0.9 at 0.1 intervals to observe the predictive performance of Drop-ResNet after training. It can be seen that the RMSE of Drop-ResNet basically shows a trend of first decreasing and then increasing. When dropout rate equals to zero, Drop-ResNet is equivalent to Normal-ResNet, and the RMSE is 6.33; when dropout rate gradually increased, such as 0.9, Drop-ResNet has the largest prediction generalization error, and the RMSE surges at 10.06, the strong regularization has dramatically destroyed the input features; when the dropout rate is properly selected, the model will strike a balance between the empirical risk and the structural risk. For example, when the dropout rate equals to 0.1, Drop-ResNet has the lowest quantization error, with the RMSE of 6.22.

Overall, Noise-ResNet outperforms Drop-ResNet with varying parameters.

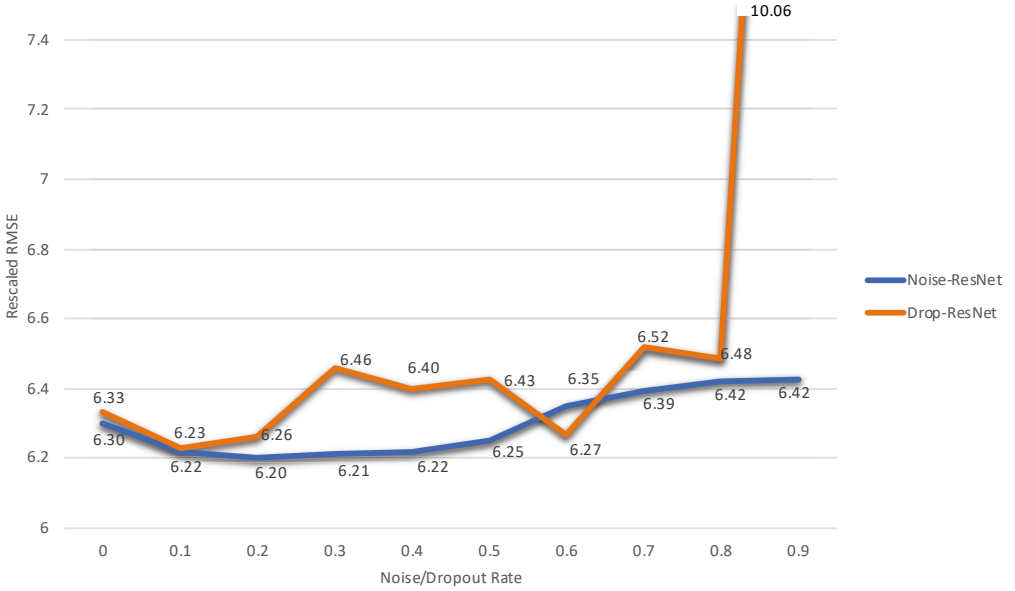


Figure 3.8 Sensitivity analysis of Noise-ResNet on BikeNYC dataset

3.5 Summary

In this chapter, we proposed a regularization method for noise perturbation of residual networks and further designed a convolutional neural network Noise-ResNet for city-wide traffic flow prediction. Experiments demonstrated that injecting the Gaussian noise into the residual connection can improve the generalization of the deep forecasting models. At the same time, it showed that incorporating various data source into the deep network may not be the best choice. By selecting the input data reasonably, it did not only improve the training speed, but also improved the forecasting performance. The single model Noise-ResNet has outperformed ST-ResNet on dataset BikeNYC . Although its performance was slightly inferior to ST-ResNet on dataset TaxiBJ, the training time of Noise-ResNet was only around 0.9 hours. Compared with the ST-ResNet training time of 7.3 hours, the saving of training time was significant. This study also demonstrated the

superior performance of the ensemble forecasting method FUSE. FUSE can greatly improve the forecasting accuracy while its training time was similar to that of a single model, and it has achieved the state-of-the-art performance on both datasets. Therefore, this study recommended that the FUSE method can be implemented as a subsequent standard step after calibrating the parameters of a single deep forecasting model.

Chapter 4

Deep Uncertainty Quantification for Weather Forecasting with Distribution Assumptions

Weather forecasting is usually solved through numerical weather prediction (NWP), which can sometimes lead to unsatisfactory performance due to inappropriate setting of the initial states. In this chapter, we design a data-driven method augmented by an effective information fusion mechanism to learn from historical data that incorporates prior knowledge from NWP. We cast the weather forecasting problem as an end-to-end deep learning problem and solve it by proposing a novel negative log-likelihood error (NLE) loss function. A notable advantage of our proposed method is that it simultaneously implements single-value forecasting and uncertainty quantification, which we refer to as *deep uncertainty quantification* (DUQ). Efficient deep ensemble strategies are also explored to further improve performance. This new approach was evaluated on a public dataset collected

from weather stations in Beijing, China. Experimental results demonstrate that the proposed NLE loss significantly improves generalization compared to mean squared error (MSE) loss and mean absolute error (MAE) loss. Compared with NWP, this approach significantly improves accuracy by 47.76%, which is a state-of-the-art result on this benchmark dataset. The preliminary version of the proposed method won 2nd place in an online competition for daily weather forecasting [1].

4.1 Introduction

Meteorological elements, such as temperature, wind and humidity, profoundly affect many aspects of human livelihood (Gneiting and Raftery, 2005; Murphy, 1993). They provide analytical support for issues related to urban computing such as traffic flow prediction, air quality analysis, electric power generation planning and so on (Zheng et al., 2014). The most common method currently utilized in meteorology is the use of physical models to simulate and predict meteorological dynamics known as numerical weather prediction, or NWP. The advantage of NWP is that it is based on the numerical solution of atmospheric hydro thermo dynamic equations and is able to obtain high prediction accuracy if the initial solution is appropriately chosen. However, NWP may not be reliable due to the instability of these differential equations (Tolstykh and Frolov, 2005). With the growing availability of meteorological big data, researchers have realized that introducing data-driven approaches into meteorology can achieve considerable success. Several machine learning methods have been applied to weather forecasting (Grover et al., 2015; Hernández et al., 2016; Sharma et al., 2011). The merit of data-driven methods is that they can quickly model patterns through learning to avoid solving

[1] *AI Challenger 2018* https://challenger.ai/competition/wf2018

complex differential equations. Nevertheless, learning from historical observations alone requires big data and a tedious amount of feature engineering to achieve satisfying performance, which presented us with the following challenge. Could we combine the advantages of NWP and machine learning to make a more efficient and effective solution? At the same time, single-value (i.e. point estimation) forecasting lacks credibility and flexibility for numerous types of human decision. Could we provide more information to indicate the prediction interval based on high-quality uncertainty quantification? This chapter aims to introduce a unified deep learning method to address these problems through end-to-end learning. In particular, we will predict multiple meteorological variables across different weather stations at multiple future steps. The proposed approach has several advantages: efficient data pre-processing, end-to-end learning, high accuracy, uncertainty quantification and easy-to-deploy which makes it have considerable practical significance. The contributions of this work are summarized as follows:

1. It proposes an effective deep model and information fusion mechanism to handle weather forecasting problems. To the best of our knowledge, this is the first machine learning method which combines historical observations and NWP for weather forecasting. Data and source codes will be released and can be used as a benchmark for researchers to study machine learning in the meteorology field.

2. It establishes effective assumptions and constructs a novel negative log-likelihood error (NLE) loss function. Unlike Bayesian deep learning (BDL), deep uncertainty quantification (DUQ) can be seamlessly integrated with current deep learning frameworks such as Tensorflow and Pytorch. It can be directly optimized via backpropagation (BP). Our experiments show that

compared with typical mean squared error (MSE) and mean absolute error (MAE) loss, training by NLE loss significantly improves the generalization of point estimation. This phenomenon has never been reported in previous researches.

3. Besides precise point estimation, DUQ simultaneously inferences the sequential prediction interval. This attractive feature has not been studied well in previous deep learning research for time series forecasting. It can be applied to various time series regression scenarios.

4. It explores efficient deep ensemble strategies. The experimental results demonstrate that the ensemble solution significantly improves accuracy.

The rest of the chapter is structured as follows. We discuss related works in Section II and introduce our method in Section III. In Section IV, we discuss experiments and performance analysis . Last, we conclude with a brief summary and shed light on valuable future works in Section VI.

4.2 Problem Statement

Let us say we have historical meteorological observations from a chosen number of weather stations and a preliminary weather forecast from NWP. For each weather station, we concern weather forecasting to approximate ground truth in the future. We define this formally below:

Notations

For a weather station s, we are given:

1. Historical observed meteorological time series
   $\mathbf{E}(t) = [e_1(t), e_2(t), ..., e_{N_1}(t)] \in \mathbb{R}^{N_1}$, where the variable e_i is one type of meteorological element, for $t = 1, ..., T_E$.

2. Another feature series consist of forecasting timesteps, station ID and NWP forecasting, i.e., $\mathbf{D}(t) = [d_1(t), d_2(t), ..., d_{N_2}(t)] \in \mathbb{R}^{N_2}$, where the variable $d_i(t)$ is one of N_2 features, for $t = T_E + 1, ..., T_E + T_D$, and T_D is the required number of forecasting steps.

3. Ground truth of target meteorological variables denoted as $\mathbf{Y}(t) = [y_1(t), y_2(t)$ $, ..., y_{N_3}(t)] \in \mathbb{R}^{N_3}$, where the variable $y_i(t)$ is one of N_3 target variables, for $t = T_E + 1, T_E + 2, ..., T_E + T_D$ and its estimation denoted as $\hat{\mathbf{Y}}(t)$.

4. Then we define:
   $\mathbf{E}_{T_E} = [\mathbf{E}(1), \mathbf{E}(2), ..., \mathbf{E}(T_E)] \in \mathbb{R}^{T_E \times N_1}$
   $\mathbf{D}_{T_D} = [\mathbf{D}(T_E + 1), \mathbf{D}(T_E + 2), ..., \mathbf{D}(T_E + T_D)] \in \mathbb{R}^{T_D \times N_2}$
   $\mathbf{X}_{T_D} = [\mathbf{E}_{T_E}; \mathbf{D}_{T_D}]$
   $\mathbf{Y}_{T_D} = [\mathbf{Y}(T_E + 1), \mathbf{Y}(T_E + 2), ..., \mathbf{Y}(T_E + T_D) \in \mathbb{R}^{T_D \times N_3}$.

Task Definition

Given $\mathbf{X}_{T_D}$, the point estimation will predict $\hat{\mathbf{Y}}_{T_D}$ to approximate $\mathbf{Y}_{T_D}$ as far as possible. The prediction interval $[\hat{\mathbf{Y}}^L_{T_D}, \hat{\mathbf{Y}}^U_{T_D}]$ will ensure $\mathbf{Y}_{T_D} \in [\hat{\mathbf{Y}}^L_{T_D}, \hat{\mathbf{Y}}^U_{T_D}]$ (element-wise) with the predefined tolerance probability. The prediction interval will cover the ground truth with at least the expected tolerance probability.

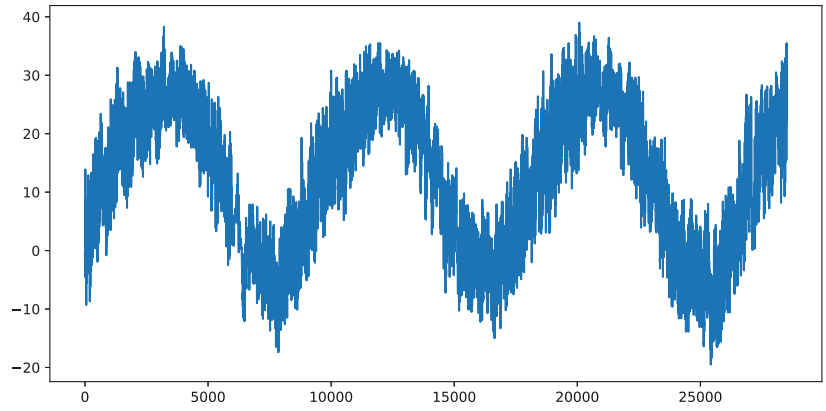
This research was driven by a real-world weather forecasting competition. For feasible comparison, it focuses on a set time period, i.e., from 3:00 intraday to 15:00 (UTC) of the next day, hence $T_D = 37$. The target variables include temperature at 2 meters (t2m), relative humidity at 2 meters (rh2m) and wind at 10 meters (w10m),

hence $N_3 = 3$. The proposed method can be easily extended for any time interval prediction and more target variables.

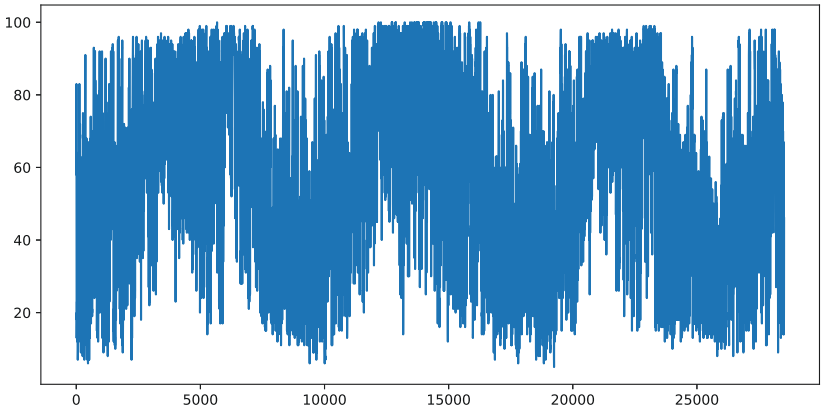
Information fusion methodology

Data exploration analysis provides insights for the motivation and methodology of information fusion. Figure 4.1 shows the variation of three target meteorological variables over the past three years. It can be seen that only temperature reflects a strong seasonal variation, while relative humidity and wind speed are subjected to much noise. Based on this observation, methods that extract seasonal features from historical data may not provide the best results, since weather changes too dramatically (Chen and Lai, 2011; Grover et al., 2015). Frequent concept drift cause long-term historical meteorological data lack value (Lu et al., 2019). One conclusion summarizes that *"For many time series tasks only a few recent time steps are required"*(Gers et al., 2002). On the other hand, NWP is a relatively reliable forecasting method, but inappropriate initial states can introduce undesirable error bias. To address this, we propose a balanced fusion methodology:

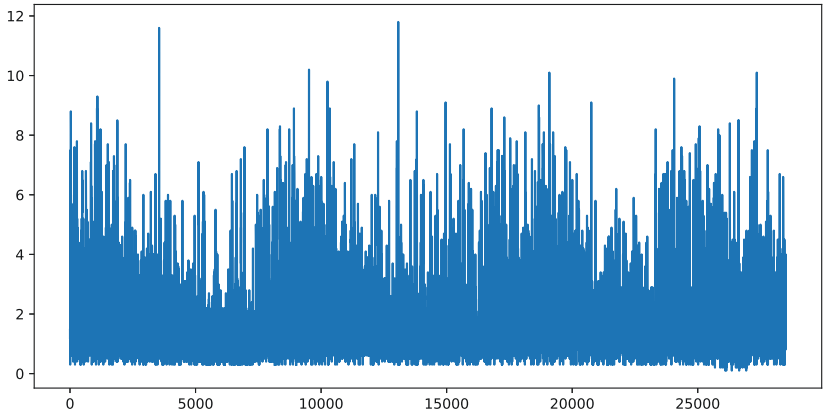
- First, *only recent* observations, i.e., $\mathbf{E}_{T_E}$ should be adopted for *modeling recent meteorological dynamics*.
- Second, a wise NWP fusion strategy should incorporate NWP forecasting *at a counterpart forecasting timestep* to easily correcting bias in NWP. Conversely, an unwise fusion strategy that is not carefully designed may absorb NWP which is not conducive to capturing important NWP signals. Hence we incorporate NWP forcasting into $\mathbf{D}_{T_D}$ rather than into $\mathbf{E}_{T_E}$ or hidden coding (see Figure 4.3).



(a) t2m



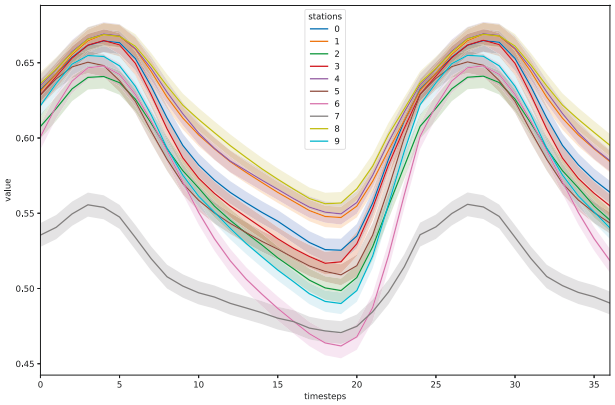
(b) rh2m



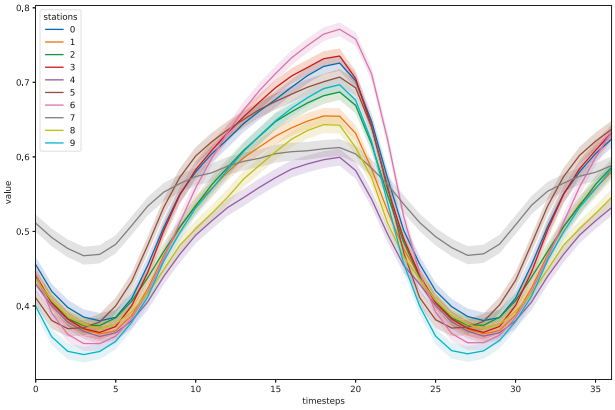
(c) w10m

Figure 4.1 Three historical target variable series from 03/01/2015-05/31/2018. They show a strong seasonal variation of t2m, a weak seasonal variation of 2-meter rh2m , and almost no seasonal variation of w10m.

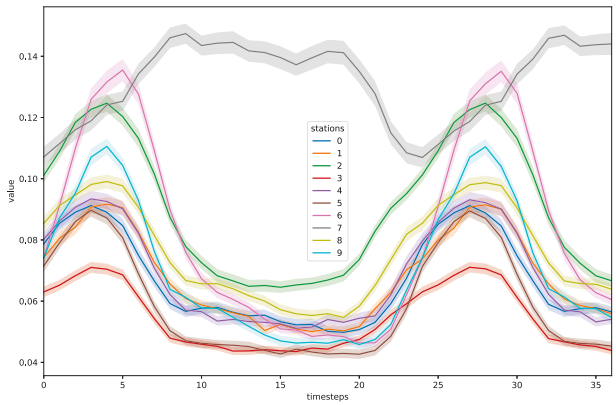
Figure 4.2 aggregates historical statistics of mean (solid line) and 90% confidence interval (shade area) for 10 stations from 3:00 intraday to 15:00 (UTC). We find that: 1) There exists obvious difference of mean and variance statistics, e.g., the mean value of station-ID 7 follows a different trend compared with other stations. 2) Every hour at every station has different meteorological characteristics of mean and variance. To address this, we will introduce station ID and time ID into $\mathbf{D}_{T_D}$.



(a) t2m



(b) rh2m



(c) w10m

Figure 4.2 The variation of mean (solid line) and 90% confidence interval (shaded area) for 10 stations during the target forecasting time zone (3:00 intraday to 15:00 of the next day) from 03/01/2015-05/31/2018. Values of Y-axis are following normalization.

4.3 Proposed Methodology

4.3.1 Data preprocessing

Missing values There are two kind of missing values, i.e. *block missing* (one-day data lost) and *local missing* (local non-continuous time series), which vary in severity. For block missing (Yi et al., 2016), we just delete the data of those days from the dataset. For local missing data, we use linear interpolation to impute missing values. Taking the training set as an example, we delete 40 days with block missing values from a total of 1188 days, leaving the training data from 1148 (1188-40) days.

Normalization of continuous variables Continuous variables without normalization sometimes result in training failure for deep leaning, so we use min-max normalization to normalize each continuous feature into [0, 1]. In the evaluation, we re-normalize the predicted values back to the normal scale.

Category variables There are two category variables, i.e. *Timesteps ID* and *Station ID*. Rather than hard-coding, such as one-hot or sin-cosine coding, we code them by embedding, which has achieved better performance than hard-coding (Mikolov et al., 2013). **Input/Output Tensors** Lastly, we load data from all stations and dates and reshape it to three tensors as follows:

- $(I, T_E, S, N_1), (I, T_D, S, N_2)$ (i.e., input tensors).
- (I, T_D, S, N_3) (i.e., ground truth tensor).

Note that I is the date index and S is the station index. When drawing training samples, we first draw integer date $i \in I$ and station $s \in S$. We can then index by i, s from these three tensors and obtain one training instance $\mathbf{X}_{T_D}^{i,s}$=$[\mathbf{E}_{T_E}^{i,s}; \mathbf{D}_{T_D}^{i,s}]$ and $\mathbf{Y}_{T_D}^{i,s}$

abbreviated as $\mathbf{X}_{T_D}$=$[\mathbf{E}_{T_E};\mathbf{D}_{T_D}]$ and $\mathbf{Y}_{T_D}$ for brevity. The advantage of organizing data in this four-tuple style is that we can conveniently index the data via the specific dimension for hindsight inspection and consideration of scalability. For example, we can index specific dates and stations for later transfer learning research. Readers can refer to the instantiated example *Parameter Settings for Reproducibility* in Section V for deeper understanding.

4.3.2 Model framework

The proposed DUQ is based on sequence-to-sequence (seq2seq, also a.k.a Encoder-Decoder). Its detailed formula is not discussed here. Readers can refer to (Srivastava et al., 2015) for more detail. There are already many high-performance variants for different tasks, but most of them focus on making improvements from the structural perspective to make point estimation more precise. We first incorporate sequential uncertainty quantification for weather forecasting into the architecture presented in Figure 4.3. The encoder first extracts latent representations $\mathbf{c}$ from the observed feature series $\mathbf{E}_{T_E}$:

$$\mathbf{c} = Enc(\mathbf{E}_{T_E};\theta_1) \tag{4.1}$$

where $\mathbf{c}$ captures the current meteorological dynamics and is then transferred to form the initial state of the decoder. Based on the memory of $\mathbf{c}$, the decoder absorbs $\mathbf{D}_{T_D}$ including station identity (StaID), forecasting time identity (TimeID), and NWP forecasting. Two embedding layers will be introduced for StaID and TimeID respectively to automatically learn the embedding representations. This architecture

will generate sequential point estimation $\mathbf{u}_{T_D}$ used as $\hat{\mathbf{Y}}_{T_D}$ to predict $\mathbf{Y}_{T_D}$ as well as the variance $\hat{\boldsymbol{\sigma}}^2_{T_D}$ utilized to estimate $[\hat{\mathbf{Y}}^L_{T_D}, \hat{\mathbf{Y}}^U_{T_D}]$:

$$\hat{\boldsymbol{\sigma}}^2_{T_D}, \hat{\mathbf{Y}}_{T_D} = Dec(\mathbf{c}, \mathbf{D}_{T_D}; \theta_2) \tag{4.2}$$

where θ_1 and θ_2 are learnable parameters. We use $f(\cdot)$ to represent the combination of Encoder-Decoder and use $\mathbf{X}_{T_D} = [\mathbf{E}_{T_E}; \mathbf{D}_{T_D}]$ which can then be regarded as:

$$\hat{\boldsymbol{\sigma}}^2_{T_D}, \hat{\mathbf{Y}}_{T_D} = f(\mathbf{X}_{T_D}) \tag{4.3}$$

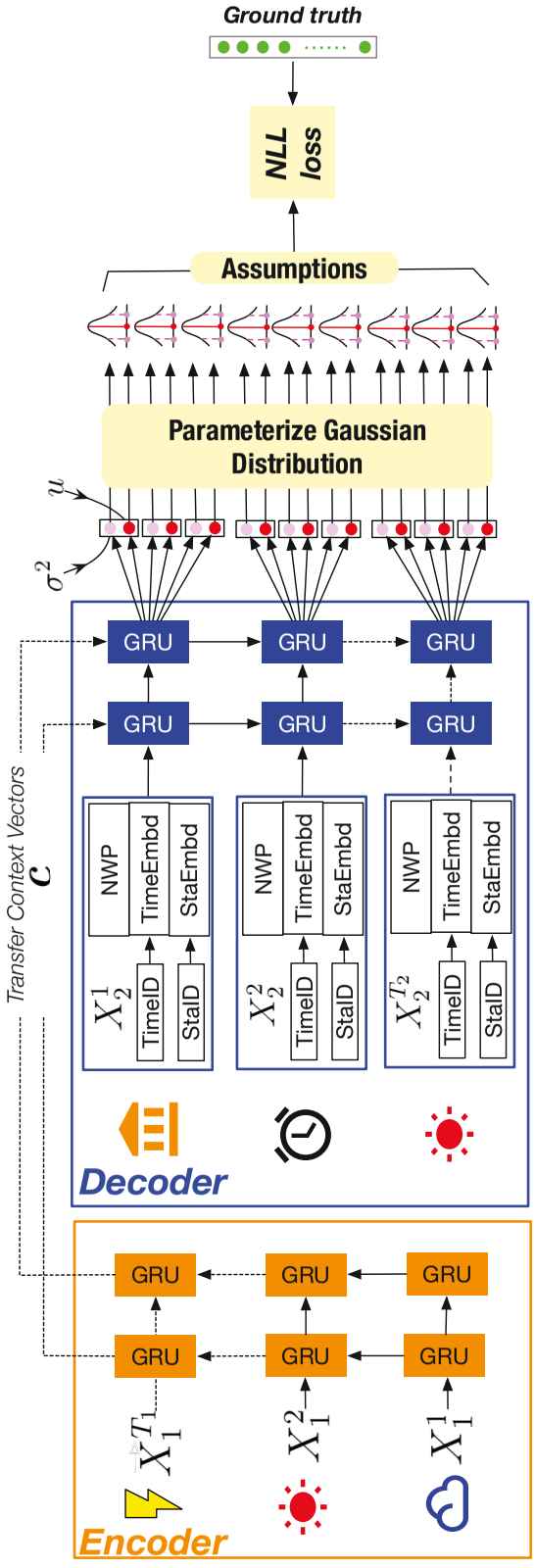


Figure 4.3 DUQ for sequential point estimation and prediction interval.

4.3.3 Likelihood loss function

Learning phase DUQ predicts two values at each timestep corresponding to the predicted mean and variance to parameterize the Gaussian distributions [2]. The NLE is calculated for the Gaussian distributions, which must be based on reasonable hypotheses. Three mild but experimentally effective assumptions are proposed (degree of effectiveness can be seen in the experimental results in Table 4.3):

1. Each day and each station are independent. This assumption ensures it is reasonable that the number of all training samples can be regard as $I \times S$. Based on this, we can minimize the negative log-likelihood error loss:

$$NLE = -\prod_{i=1}^{I}\prod_{s=1}^{S} p_\theta(\mathbf{Y}_{T_D}|\mathbf{X}_{T_D}) \tag{4.4}$$

2. Each target variable and each timestep at one specific station are *conditionally independent* given $\mathbf{X}_{T_D}$. Based on this, we can further decompose $p_\theta(\mathbf{Y}_{T_D}|\mathbf{X}_{T_D})$ by the product rule and transform it via log operation as:

$$p_\theta(\mathbf{Y}_{T_D}|\mathbf{X}_{T_D}) = \prod_{o=1}^{N_3}\prod_{t=1}^{T_D} p_\theta(y_o(t)|\mathbf{X}_t) \tag{4.5}$$

$$log\ p_\theta(\mathbf{Y}^i_{T_D}|\mathbf{X}^i_{T_D}) = \sum_{o=1}^{N_3}\sum_{t=1}^{T_D} log\ p_\theta(y_o(t)|\mathbf{X}^i_t) \tag{4.6}$$

3. The target variables satisfy multivariate independent Gaussian distribution and σ_θ is a function of the input features, i.e., $\mathbf{Y}_{T_D} \sim N(u_\theta(\mathbf{X}_{T_D}), \sigma_\theta(\mathbf{X}_{T_D}))$.

[2] We enforce the positivity constraint on the variance by passing the second output through the $softplus$ function $log(1+exp(\cdot))$, and add a minimum variance (e.g. 10^{-6}) for numerical stability.

Based on this assumption, the final loss is:

$$\begin{aligned} NLE &= -\sum_{i}^{I}\sum_{o=1}^{N_3}\sum_{t=1}^{T_D} log\ p_\theta(y_o(t)|\mathbf{X}_t^i) \\ &= \sum_{i}^{I}\sum_{o=1}^{N_3}\sum_{t=1}^{T_D} \frac{log\sigma_{o;\theta}^2(\mathbf{X}_t^i)}{2} + \frac{(y_o(t)-u_{o;\theta}(\mathbf{X}_t^i))^2}{2\sigma_{o;\theta}^2(\mathbf{X}_t^i)} + C \end{aligned} \tag{4.7}$$

where C is a constant which can be omitted during training. $y_o(t)$ is the ground truth of a target variable o at timestep t, $\sigma_{o;\theta}^2(\mathbf{X}_t^i)$ and $u_{o;\theta}(\mathbf{X}_t^i)$ are respectively the variance and mean of a Gaussian distribution parameterized by DUQ. The aim of the entire learning phase is to minimize NLE. Optimizing by deep models can easily lead to overfitting on the training set, therefore it is necessary to implement early-stopping on the validation set. Algorithm 4.1 outlines the procedure for the learning phase.

Inference phase

After training, we can implement statistical inference for an input $\mathbf{X}_{T_D}$ by:

$$u_\theta(\mathbf{X}_{T_D}), \sigma_\theta^2(\mathbf{X}_{T_D}) = f(\mathbf{X}_{T_D}) \tag{4.8}$$

where $u_\theta(\mathbf{X}_{T_D})$ is statistically the mean estimation i.e., $\hat{\mathbf{Y}}_{T_D}$ given $\mathbf{X}_{T_D}$, which will be adopted for forecasting and $\sigma_\theta^2(\mathbf{X}_{T_D})$ is statistically the variance estimation, i.e., $\hat{\boldsymbol{\sigma}}_{T_D}^2$ given $\mathbf{X}_{T_D}$. Recall our assumption that $\hat{\mathbf{Y}}_{T_D}$ satisfies Gaussian distribution, so upper bound $\hat{\mathbf{Y}}_{T_D}^U$and lower bound $\hat{\mathbf{Y}}_{T_D}^L$ can be inferenced as follows:

$$\hat{\mathbf{Y}}_{T_D}^U = \hat{\mathbf{Y}}_{T_D} + \lambda\hat{\boldsymbol{\sigma}}_{T_D} \tag{4.9}$$

$$\hat{\mathbf{Y}}_{T_D}^L = \hat{\mathbf{Y}}_{T_D} - \lambda\hat{\boldsymbol{\sigma}}_{T_D} \tag{4.10}$$

Algorithm 4.1: Algorithm for learning

Input : N_1, N_2, N_3, T_E, T_D;
Input tensors: (I, T_E, S, N_1), (I, T_D, S, N_2);
Output tensor: (I, T_D, S, N_3);
Maximum iterations;
Tolerance iterations for early-stopping;

Output : Learned DUQ model

// Learning phase

Initialize all learnable parameters θ in DUQ
repeat
  $\mathcal{B} \leftarrow \emptyset$
  while *each training datum n (*$1 \leq n \leq BatchSize$*)* **do**
    // Format training data samples
    Draw a random integer $i \sim uniform(0, I)$
    Draw a random integer $s \sim uniform(0, S)$
    Index by i, s from input and output tensors and get one training sample $(\mathbf{X}_{T_D}, \mathbf{Y}_{T_D})$
    Put this sample into $\mathcal{B}$
  end
  Update θ via BP by minimizing the NLE loss on $\mathcal{B}$
until *stopping criteria are met*;

where $\hat{\boldsymbol{\sigma}}_{T_D}$ is the standard deviation and, λ should be determined according to the pre-defined $1-z$. In this research, $1-z=0.9$ thus λ is set to 1.65 according to the z-score of Gaussian distribution. Algorithm 4.2 gives the inference procedure.

Algorithm 4.2: Algorithm for inference

Input : $\mathbf{X}_{T_D}$, z;
Output : $\hat{\mathbf{Y}}^{L}_{T_D}, \hat{\mathbf{Y}}^{U}_{T_D}, \hat{\mathbf{Y}}_{T_D}, \hat{\boldsymbol{\sigma}}_{T_D}$;
// Inference phase
$\hat{\mathbf{Y}}_{T_D}, \hat{\boldsymbol{\sigma}}_{T_D} = f(\mathbf{X}_{T_D})$
// determine λ by z from z-score of Gaussian distribution
$\hat{\mathbf{Y}}^{L}_{T_D} = \hat{\mathbf{Y}}_{T_D} - \lambda \hat{\boldsymbol{\sigma}}_{T_D}$
$\hat{\mathbf{Y}}^{U}_{T_D} = \hat{\mathbf{Y}}_{T_D} + \lambda \hat{\boldsymbol{\sigma}}_{T_D}$
de-normalize $\hat{\mathbf{Y}}^{L}_{T_D}, \hat{\mathbf{Y}}^{U}_{T_D}, \mathbf{Y}_{T_D}$ for real-world evaluation.

Ensemble methodology We adopt a simple but efficient principle for ensemble: each single model is a DUQ-based model initialized with specified nodes. The ensemble point estimation is the averaged point estimation of all DUQ-based models, which is scalable and easily implementable.

Point Estimation Measurement We first calculate the root mean squared error (*RMSE*) for each objective variable from all stations for daily evaluation.

$$RMSE_{obj} = \sum_{s=1}^{S} ||\mathbf{Y}^{s,obj}_{T_D} - \hat{\mathbf{Y}}^{s,obj}_{T_D}|| \tag{4.11}$$

where $\mathbf{Y}^{s,obj}_{T_D}$ and $\hat{\mathbf{Y}}^{s,obj}_{T_D}$ are respectively the ground truth and the predicted value of the objective variable (i.e., $t2m$, $rh2m$ or $w10m$ in this chapter) at station s.

$$RMSE_{day} = \frac{RMSE_{t2m} + RMSE_{rh2m} + RMSE_{w10m}}{N_3} \tag{4.12}$$

$RMSE_{day}$ is the ultimate RMSE criterion in the experimental reports for each day. $RMSE_{avg}$ is the average $RMSE_{day}$ over all days. To demonstrate the improvement

over the classic NWP method, we employ the following evaluation using the associated skill score (*SS*, the higher the better):

$$SS_{obj} = 1 - \frac{RMSE_{obj\_ml}}{RMSE_{obj\_nwp}} \tag{4.13}$$

where $RMSE_{obj\_nwp}$ is the $RMSE_{obj}$ calculated by the NWP method and $RMSE_{obj\_ml}$ is calculated from the prediction made of machine learning models.

$$SS_{day} = \frac{SS_{t2m} + SS_{rh2m} + SS_{w10m}}{N_3} \tag{4.14}$$

SS_{day} is the ultimate *SS* criterion in experimental reports for every day. SS_{avg} is the average SS_{day} over all days, which is also the ultimate rank score in the online competition.

Prediction Interval Measurement

To evaluate the prediction interval, we introduce the metric called prediction interval coverage probability (PICP). First, an indicator vector $\mathbf{B}(t) = [b_1(t), b_2(t), ..., b_{N_3}(t)] \in \mathbb{R}^{N_3}$ is defined, for $t = T_E + 1, ..., T_E + T_D$. Each Boolean variable $b_o(t) \in [0,1]$ represents whether the objective variable o at the predicted time step t has been captured by the estimated prediction interval.

$$b_o(t) = \begin{cases} 1, & \text{if} \quad \hat{y}_o^L(t) \leq y_o(t) \leq \hat{y}_o^U(t) \\ 0, & \text{else.} \end{cases} \tag{4.15}$$

The total number of captured data points for the objective variable is defined as C_{obj},

$$C_{obj} = \sum_{s=1}^{S} \sum_{t=T_E+1}^{T_E+T_D} b_o(t)$$

Then $PICP_{obj}$ for the objective variable is defined as:

$$PICP_{obj} = \frac{C_{obj}}{T_D * S} \tag{4.16}$$

Ideally, $PICP_{obj}$ should be equal to or greater than the pre-defined value, i.e., $1 - z = 0.9$ where z is the significant level and is set to 0.1 in our experiments.

4.4 Experiments and Analysis

4.4.1 Baselines

SARIMA (Fang and Lahdelma, 2016) Seasonal autoregressive integrated moving average is a benchmark model for univariate time series, where parameters are chosen using AIC (Akaike information criterion).

SVR (Chen et al., 2017) Support vector regression is a non-linear support vector machine for regression estimation.

GBRT (Persson et al., 2017) Gradient boosting regression tree is an ensemble method for regression tasks and is widely used in practice.

DUQ$_{50}$ is one layer GRU-based seq2seq with 50 hidden nodes. The loss function is *NLE*.

DUQ$_{50-50}$ is two layers GRU-based seq2seq with 50 hidden nodes of each layer. The loss function is *NLE*.

DUQ$_{200}$ is one layer GRU-based seq2seq with 200 hidden nodes. The loss function is *NLE*.

DUQ$_{300-300}$ is two layers GRU-based seq2seq with 300 hidden nodes of each layer. The loss function is *NLE*.

DUQ$_{noNWP}$ is the same as DUQ$_{300-300}$ except that NWP forecasting (i.e. **NWP** of $\mathbf{D}_{T_D}$, refer to Figure 4.3) is masked by zero values.

DUQ$_{noOBS}$ is the same as DUQ$_{300-300}$ except that the observation features (i.e. $\mathbf{E}_{T_E}$) are masked by zero values.

Seq2Seq$_{MSE}$ is the same as DUQ$_{300-300}$ except that the loss function is MSE.

Seq2Seq$_{MAE}$ is the same as DUQ$_{300-300}$ except that the loss function is MAE.

DUQ$_{Esb3}$ ensembles three DUQ models (i.e., DUQ$_{300-300}$, DUQ$_{200-200}$, DUQ$_{100-100}$) for online evaluation. This method achieved 2nd place in the online competition.

DUQ$_{Esb10}$ ensembles 10 DUQ models with different architecture to explore the effectiveness of the ensemble. It ensembles DUQ$_{300-300}$, DUQ$_{310-310}$, ..., DUQ$_{390-390}$ (increasing at 10-neuron intervals).

Model$_{1st}$ achieves the best SS_{avg} during online comparison. According to the on-site report, the author also adopted a complicated stacking and ensemble learning strategy.

4.4.2 Experimental environments

The experiments were implemented on a GPU server with Quadro P4000 GPU and Keras programming environment (Tensorflow backend).

4.4.3 Parameter settings

The *batch size* is set to 512. The *embedding dimension* of each embedding layer is set to 2. Since we adopted an early-stopping strategy, it was not necessary to set the *epoch* parameter. Instead, we set the number of *maximum iterations* to a relatively large number of 10000 to take sufficient batch iterations into consideration.

The *validation interval* (*vi*) is set to 50 meaning that for every 50 iterations, we will test our model on validation set and calculate the validation loss. We set the *early-stopping tolerance* (*est*) to 10, meaning that if the validation loss over 10 continuous iterations did not decrease, training would be stopped early. We defined the *validation times* (*vt*) when early-stopping was triggered, hence the *total iterations* (*ti*) can be calculated by $ti = vt \times vi$. For the prediction interval, z was set to 0.1, $1 - z = 0.9$ thus λ is set to 1.65 according to the z-score of Gaussian distribution.

We set $N_1 = 9, N_2 = 31, N_3 = 3, T_E = 28, T_D = 37$ to preprocess the original dataset. After preprocessing, the final dataset shape was as shown below.

For the training set:

- Encoder inputs: (1148, 28, 10, 9)
- Decoder inputs: (1148, 37, 10, 31)
- Decoder outputs: (1148, 37, 10, 3)

For the validation set:

- Encoder inputs: (87, 28, 10, 9)
- Decoder inputs: (87, 37, 10, 31)
- Decoder outputs: (87, 37, 10, 3)

For the test set on each day:

- Encoder inputs: (1, 28, 10, 9)
- Decoder inputs: (1, 37, 10, 31)

- Decoder outputs: (1, 37, 10, 3)

The meaning of each number is explained as follows: we acquired data from 1148 days for the training set and data from 87 days for the validation set. Because our evaluation is based on online daily forecasting, the test day index is 1. Number 28 is a hyperparameter, meaning that the previous 28 hours of observations were used to model recent meteorological dynamics. Number 37 was set according to the specified forecasting steps for the next 37 hours. Number 9 is the dimension of observed meteorological variables. Number 31 (dimension of decoder inputs) consists of concatenating *Timesteps ID* and *Station ID* into 29-dimension of NWP forecasting (2+29=31). Number 3 is the ground truth number for 3 target variables. The size of the final training set is 1148*10=11480. The size of validation set is 87*10=870, which is used for early-stopping. The size of test set on each day is 1*10=10.

4.4.4 Performance analysis

Table 4.1 presents the evaluation by *SS* score based on rolling forecasting, with incremental data releasd on a daily basis for nine days to mimic real-world forecasting processes.

Effect of information fusion Comparing $\text{DUQ}_{300-300}$ with DUQ_{noNWP} validates the effectiveness of fusing NWP forecasting. Comparing $\text{DUQ}_{300-300}$ with DUQ_{noOBS} validates the effectiveness of modeling recent meteorological dynamics.

Effect of deep learning On average, the deep learning-based models (DUQ and Seq2Seq) perform better than the non-deep learning models (SARIMA, SVR, GBRT). Comparing DUQ_{50} and DUQ_{50-50} validates the influence of deeper layers.

Comparing DUQ_{50}, DUQ_{200}, and $\text{DUQ}_{300-300}$ validates the effectiveness of nodes under the same number of layers.

Effect of loss function A notable result is that $\text{DUQ}_{300-300}$ trained by NLE loss performs much better than Seq2Seq_{MSE} (MSE loss) and Seq2Seq_{MAE} (MAE loss). In order to empirically understand the reasons for better generalization when trained by NLE, we calculated the ti when early-stopping was triggered, as shown in Table 4.5. It can be seen that $\text{DUQ}_{300-300}$ requires more iterations to converge. A reasonable interpretation is that NLE loss jointly implements two tasks i.e., mean optimization and variance optimization, which need more iterations to converge. This joint optimization may to some extent play a regularization role and help each other out of the local minimum. It may therefore require more iterations to converge and may have better generalization. We believe that this phenomenon deserves the attention of researchers, and that it should be proved by theory in follow-up work.

Effect of ensemble The ensemble model DUQ_{Esb3} was used in the online competition [3]. DUQ_{Esb10} achieved the best SS_{avg}, which indicates that ensemble with more DUQ models would provide a better solution.

Significance of T-test Because real-world forecasting only covers nine days, we implemented a *one-tail paired T-test* with significance level $sig = 0.25$ to ensure that our results were statistically significant. The column *P-value* between DUQ_{Esb10} and others shows that each T-test has been passed which means that our method DUQ_{Esb10} is significantly better than any other baseline under the specified significance level. For the single model, we also implemented a T-test between

[3]Readers can refer to https://challenger.ai/competition/wf2018 to check our online scores which are consistent with DUQ_{Esb3} during Day 3-Day 9. During Day 1 and Day 2 of the online competition, DUQ_{Esb3} had not been developed. In this chapter, we re-evaluate DUQ_{Esb3} offline on Day 1 and Day 2. This is also why DUQ_{Esb3} with SS_{avg}=0.4673 is better than the Model_{1st} (0.4671) but was only awarded 2nd place online.

$\text{DUQ}_{300-300}$ and other baselines including DUQ_{noOBS}, DUQ_{noNWP}, Seq2Seq_{MSE}, Seq2Seq_{MAE} to ensure that $\text{DUQ}_{300-300}$ had significant effectiveness, which is shown in Table. 4.4.

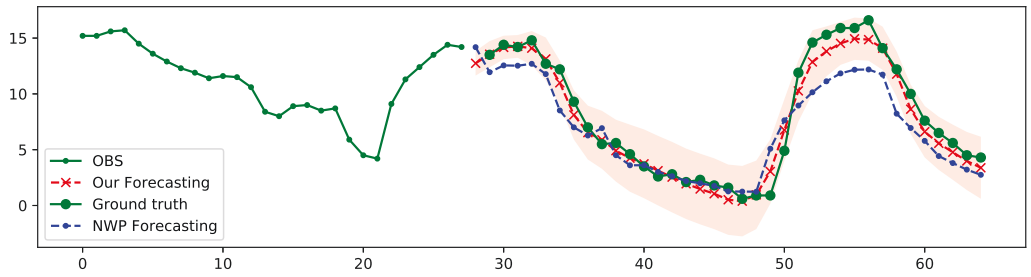
Evaluation by RMSE We also evaluated all methods by $RMSE_{avg}$ as shown in Table 4.2. Since $RMSE_{avg}$ and SS_{avg} do not have a fully linear relationship, the counterpart assessment does not reach the optimum at the same time while DUQ_{Esb10} still achieves the best $RMSE_{avg}$. The related P-value also indicates that DUQ_{Esb10} is significantly better than other baselines under $sig = 0.25$. Because online evaluation does not release the RMSE ranking, the RMSE of Model_{1st} place is not shown in Table 4.2.

Discuss instability of weather forecasting Due to meteorological instability and variability, no single model can achieve the best scores every day - not even the ensemble method. Sometimes a single model can achieve the highest SS_{day} score, such as DUQ_{200} on Day 2 and DUQ_{50-50} on Day 7. Overall, however, the ensemble method DUQ_{Esb10} achieves the greatest score benefit from the stability of ensemble learning. The instability of meteorological elements also reflects the need for a prediction interval.

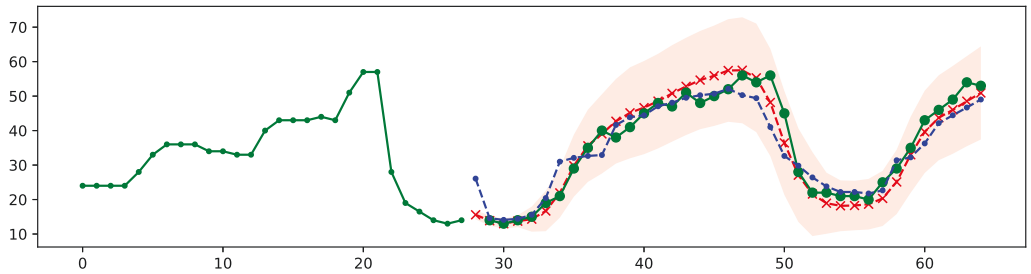
Quantity of prediction interval An effective prediction interval should satisfy that $PICP_{obj}$ is equal to or greater than the pre-defined $1-z = 90\%$. Table 4.3 shows the results. In particular, the $PICP_{rh2m}$ on Day 3 seems far below expectations. The main reason is that forecasting of *rh2m* is not accurate on that day. The online competition scores of all contestants were particular low on that day. Generally, our approach meets the requirement $PICP_{avg} \geq 1-z = 90\%$.

Quality of prediction interval We take the model $\text{DUQ}_{300-300}$ to visualize the quality of the prediction interval. Figure 4.4 illustrates a forecasting instance at one

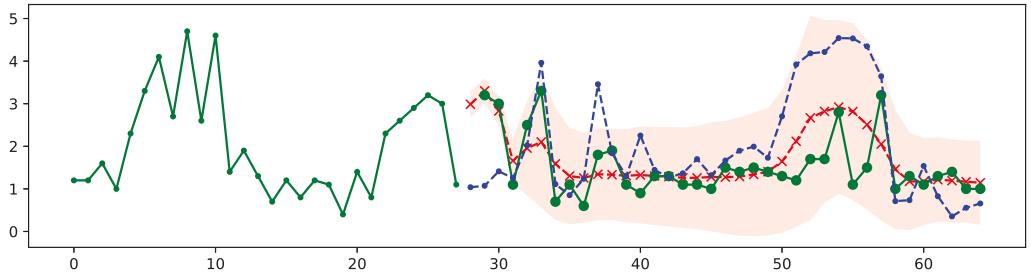
station on a competition day. In each sub-figure, the left green line is the observed meteorological value during the previous 28 hours, the right green line is the ground truth, the blue line is the NWP prediction, the red line is $DUQ_{300-300}$ prediction and the red shaded area is the 90% prediction interval. A noticeable observation is that the prediction interval does not become wider over time, instead, it presents that the width of the middle part is narrower than both ends particularly for t2m and rh2m.A reasonable explanation is that meteorological elements largely change during the daytime and become more stable during night time. Having this prediction interval would provide more information for travel/production planning than only point prediction. Another noteworthy point is that because w10m fluctuates sharply, it is more difficult to forecast point estimate precisely, and the prediction interval tends to be wider than t2m and rh2m.



(a) t2m



(b) rh2m



(c) w10m

Figure 4.4 A test sample at one station is chosen to visualize the forecasting of 3 target variables in the future 37 hours. We can see that all predicted points fall into the prediction interval given by $1-z=90\%$.

Table 4.1 The *SS* performance of different methods on 9 days. The column P-value compare the best performing method DUQ_{Esb10} with other methods, using one-tail paired T-test.

Method	SS_{day1}	SS_{day2}	SS_{day3}	SS_{day4}	SS_{day5}	SS_{day6}	SS_{day7}	SS_{day8}	SS_{day9}	SS_{avg}	P-value
SARIMA	0.1249	-1.4632	-0.2417	-0.4421	-0.2631	-0.2301	0.0630	0.2015	-0.4579	-0.3010	0.00
SVR	-0.7291	-0.6342	-0.1999	-0.5918	-1.1230	-0.8568	-0.6154	-0.5123	-0.5807	-0.6492	0.00
GBRT	0.0221	0.1318	-0.0086	-0.0396	-0.0960	0.0067	0.0772	0.0859	0.0000	0.0199	0.00
DUQ_{50}	0.4813	0.4833	0.2781	0.3053	0.4277	0.4853	0.4609	0.4987	0.2647	0.4095	0.00
DUQ_{50-50}	0.4847	0.4969	0.3088	0.4012	0.4302	0.5051	**0.5656**	0.5502	0.3239	0.4518	0.00
DUQ_{200}	0.5278	**0.5088**	0.2890	0.3797	0.4479	0.5358	0.4961	0.5235	0.3478	0.4507	0.02
$\text{DUQ}_{300-300}$	0.5220	0.5002	0.3352	0.4067	0.4474	0.5289	0.5324	0.5463	0.3047	0.4582	0.00
DUQ_{noNWP}	0.2348	0.2992	0.0081	0.2440	0.1630	0.3125	0.2660	0.3003	-0.1599	0.1853	0.00
DUQ_{noOBS}	0.4694	0.4744	0.2624	0.3447	0.3925	0.4588	0.4756	0.4901	0.3150	0.4092	0.00
Seq2Seq_{MSE}	0.4978	0.3934	0.2860	0.3960	0.3965	0.4842	0.4820	0.5138	0.3192	0.4188	0.00
Seq2Seq_{MAE}	0.5314	0.4346	0.2671	0.3980	0.4610	0.5391	0.4711	0.5565	0.2999	0.4399	0.00
DUQ_{Esb3} (2rd place)	0.5216	0.4951	0.3358	0.4050	0.4627	0.5359	0.5350	0.5664	**0.3479**	0.4673	0.04
DUQ_{Esb10}	**0.5339**	0.4940	**0.3516**	0.4355	0.4600	0.5575	0.5581	**0.5776**	0.3298	**0.4776**	-
Model_{1st}	0.4307	0.4847	0.3088	**0.4572**	**0.5019**	**0.5753**	0.5345	0.5726	0.3384	0.4671	0.24

Table 4.2 The *RMSE* performance of different methods on 9 days. Since *RMSE* and *SS* are not fully linear relationship, the counterpart assessment does not reach the optimal at the same time.

Method	$RMSE_{day1}$	$RMSE_{day2}$	$RMSE_{day3}$	$RMSE_{day4}$	$RMSE_{day5}$	$RMSE_{day6}$	$RMSE_{day7}$	$RMSE_{day8}$	$RMSE_{day9}$	$RMSE_{avg}$	P-value
NWP	7.5923	9.7276	5.1079	5.7335	6.1542	7.5239	6.3647	7.0457	5.8819	6.7924	0.00
SARIMA	7.3017	16.5954	7.2964	8.9968	8.0726	8.1440	7.1722	5.8466	8.1795	8.6228	0.00
SVR	8.1788	10.0436	5.9310	7.1662	7.7576	8.4064	7.9094	8.4749	7.5431	7.9346	0.00
GBRT	6.5551	8.0321	5.6892	6.1422	6.1083	6.5687	5.9449	6.7122	5.9573	6.4122	0.00
DUQ_{50}	3.0432	4.5256	4.1858	4.5445	3.0069	3.4912	3.8087	3.2299	4.5545	3.8211	0.00
DUQ_{50-50}	2.9013	4.2442	4.0203	3.6641	3.2275	3.2062	**2.6428**	2.8175	4.3089	3.4481	0.00
DUQ_{200}	2.8128	**4.0400**	4.1624	4.0732	2.8580	2.8086	3.2870	3.1963	4.3326	3.5079	0.03
$DUQ_{300-300}$	2.7168	4.0615	3.8866	3.7977	2.8083	2.9211	2.8012	2.9784	**4.3308**	3.3669	0.16
DUQ_{noNWP}	5.0371	5.5370	5.0529	4.6819	4.1385	4.4716	5.7058	5.8346	7.6805	5.3489	0.00
DUQ_{noOBS}	3.2170	4.8604	4.4150	4.1303	3.5896	3.8239	3.4992	3.2031	4.3008	3.8933	0.00
$Seq2Seq_{MSE}$	3.1328	5.0769	4.1400	3.8426	3.2040	3.3142	3.3027	3.1785	4.6426	3.7594	0.00
$Seq2Seq_{MAE}$	2.7272	5.0933	4.2837	4.0184	2.7888	3.0029	3.7165	2.7935	4.6509	3.6750	0.00
DUQ_{Esb3}	2.8000	4.4338	**3.7054**	3.7886	2.8566	2.7890	2.7979	2.8011	4.3310	3.3670	0.05
DUQ_{Esb10}	**2.7027**	4.3341	3.7999	**3.5743**	**2.7627**	**2.6874**	2.7799	**2.7402**	4.3949	**3.3085**	-

Table 4.3 PICP on every day

$PICP_{obj}$	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	$PICP_{avg}$
$PICP_{t2m}$	0.9513	0.9351	0.8918	0.8891	0.9594	0.9648	0.9027	0.8945	0.9351	**0.9249**
$PICP_{rh2m}$	0.9945	0.8702	0.7648	0.8729	0.9621	0.9621	0.9243	0.9243	0.9135	**0.9099**
$PICP_{w10m}$	0.9567	0.9567	0.9081	0.9594	0.9675	0.9621	0.9378	0.9648	0.9540	**0.9519**

Table 4.4 T-test among single models for $DUQ_{300-300}$

	P-value on $RMSE_{avg}$	P-value on SS_{avg}
$DUQ_{300-300}$	-	-
DUQ_{noNWP}	0.00	0.00
DUQ_{noOBS}	0.00	0.00
$Seq2Seq_{MSE}$	0.00	0.00
$Seq2Seq_{MAE}$	0.03	0.08

Table 4.5 Iterations

Methods	ti (vt$\times$ vi)
$DUQ_{300-300}$	2900 (58*50)
$Seq2Seq_{MSE}$	2100 (42*50)
DUQ_{noNWP}	1950 (39*50)
$Seq2Seq_{MAE}$	1850 (37*50)
DUQ_{noOBS}	1450 (29*50)

4.5 Summary

This chapter addressed the real-world problem in weather forecasting and introduced a new deep uncertainty quantification (DUQ) method. A novel loss function called negative log-likelihood error (NLE) was designed to train the prediction model, which was capable of simultaneously inferring sequential point estimation and prediction interval. A noteworthy experimental serendipity was that training by NLE loss significantly improved the generalization of point estimation. This may provide practitioners with new insights to develop and deploy learning algorithms for forcasting and regression problems. Based on the proposed method and an efficient deep ensemble strategy, it achieved state-of-the-art performance on a real-world weather forecasting benchmark dataset. The overall method was developed in Keras and was flexible to be deployed in the production environment. The source codes has been released and can be used as a benchmark for researchers and practitioners to investigate weather forecasting problems. Future works will be directed towards architecture improvement (e.g., attention mechanism), automatic hyperparameter-tuning, and theoretical comparison between NLE and MSE/MAE.

Chapter 5

Deep Uncertainty Quantification without Distribution Assumptions

Time series forecasting is a challenging task as the underlying data generating process is dynamic, nonlinear, and uncertain. Deep learning such as LSTM and auto-encoder can learn representations automatically and has attracted considerable attention in time series forecasting. However, current approaches mainly focus on point estimation, which leads to the inability to quantify uncertainty. Meantime, existing deep uncertainty quantification methods suffer from various limitations in practice. To this end, this chapter presents a novel end-to-end framework called deep prediction interval and point estimation (DeepPIPE) that simultaneously performs multi-step point estimation and uncertainty quantification for time series forecasting. The merits of this approach are threefold: first, it requires no prior assumption on the distribution of data noise; second, it utilizes a novel hybrid loss function that improves the accuracy and stability of forecasting; third, it is only optimized by back-propagation algorithm, which is time friendly and easy

to be implemented. Experimental results demonstrate that the proposed approach achieves state-of-the-art performance on three real-world datasets.

5.1 Introduction

Time series are observations of a dynamic system collected sequentially over time (Choudhary et al., 2018). To forecast the future values, historical t-step observations $\mathbf{X}_{1:t} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t] \in R^{t \times d}$ with each $\mathbf{x}_i \in R^d$ and d being the feature dimensions, are analyzed to build a model that depicts the underlying dynamic of the nonlinear system. The model is then utilized to predict the most possible series $\hat{\mathbf{y}}_{t+1:t+S} = [\hat{y}_{t+1}, \hat{y}_{t+2}, ..., \hat{y}_{t+S}] \in R^S$ to approximate the ground truth series $\mathbf{y}_{t+1:t+S}$ in the future S steps, which can be formally described as below:

$$\hat{\mathbf{y}}_{t+1:t+S} = \underset{\mathbf{y}_{t+1:t+S}}{\operatorname{argmax}} Pr(\mathbf{y}_{t+1:t+S} | \mathbf{X}_{1:t}) \tag{5.1}$$

or denoted by the machine learning model $f(\cdot)$:

$$\hat{\mathbf{y}}_{t+1:t+S} = f(\mathbf{X}_{1:t}) \tag{5.2}$$

The forecasted $\hat{\mathbf{y}}_{t+1:t+S}$ is termed as *point estimation*, a.k.a, single-value forecasting. Previous studies of deep learning for time series forecasting have achieved great success, especially on point estimation problems (Guo et al., 2018; Li et al., 2018; Wang et al., 2016; Yu et al., 2017; Zhang et al., 2018, 2017b). Nonetheless, point estimation omits uncertainty, which cannot provide sufficient information for safe and optimal decision-making (Zeng and Lu, 2018). For example, crowd prediction and management without uncertainty quantification will lack warning information

for managers to take early deployment measures (Helbing et al., 2015). In financial trading, if a prediction algorithm only forecasts a single value, it will not be sufficient to derive a safe trading policy in that the market can be high-risk and volatile (Zhang et al., 2017b). Concerning many automated systems, various safety issues may arise beyond expectation (Perc et al., 2019), hence letting the automated systems sense high uncertain scenarios to stop operations and seek experts for intervention will be the safe strategy. *Uncertainty quantification* has therefore received increased attention in the research community (Khosravi et al., 2011a; Yeo et al., 2018). Through the uncertainty quantification method, researchers proposed LSTM-based auto-encoder scheme for anomaly detection (Malhotra et al., 2016). In order to formalize uncertainty quantification in the scenarios of time series forecasting, let us first consider the equation of one-step point estimation by setting $n = 1$ in formula (5.2) and it achieves:

$$\hat{y}_{t+1} = f(\mathbf{X}_{1:t}) \tag{5.3}$$

Although $\hat{y}_{t+1}$ is regarded as the prediction for the ground truth y_{t+1}, a basic view of statistical machine learning is that y_{t+1} is difficult to be forecasted perfectly and there often exists irreducible noise ε_{t+1} in y_{t+1}, which is expressed as:

$$y_{t+1} = f(\mathbf{X}_{1:t}) + \varepsilon_{t+1} \tag{5.4}$$

The philosophy of uncertainty quantification is to predict a prediction interval (PI) $[\hat{y}^L_{t+1}, \hat{y}^U_{t+1}]$ to bound y_{t+1} to satisfy

$$Pr_{t+1} = Pr[\hat{y}^L_{t+1} \leq y_{t+1} \leq \hat{y}^U_{t+1}] \geq P_c, \tag{5.5}$$

where P_c is the predefined confidence level and Pr_{t+1} is called prediction interval coverage probability (PICP). It can be inferred that if $\hat{y}^U_{t+1}$ is extremely large and in the meantime $\hat{y}^L_{t+1}$ is excessively small, Pr_{t+1} can be always equal to 100%, which is nonsensical for users. An admirable uncertainty quantification is when formula (5.5) holds, smaller interval width, i.e., $\hat{y}^U_{t+1} - \hat{y}^L_{t+1}$, is better. The equations (5.3), (5.4) and (5.5) can be extended for the other time step $t+S$.

This study aims to utilize one deep learning model to forecast multi-step point estimation and prediction interval simultaneously in time series scenarios. The contributions of this chapter are summarized as below:

1. It proposes an end-to-end deep learning method to solve sequential point estimation and uncertainty quantification simultaneously for time series forecasting. More formally, it is an integrated model $g(\cdot)$ with the specified P_c to predict $\hat{\mathbf{y}}, \hat{\mathbf{y}}^L$ and $\hat{\mathbf{y}}^U \in R^S$, i.e., $\hat{\mathbf{y}}_{t+1:t+S}, \hat{\mathbf{y}}^L_{t+1:t+S}, \hat{\mathbf{y}}^U_{t+1:t+S} = g(\mathbf{X}_{1:t}|P_c)$.

2. It is distribution-free for modeling ε_{t+1}, which means it implicitly learns the distribution of the irreducible noise ε_{t+1} instead of imposing an explicit distribution assumption. This property makes it very consistent with the real-world environment where ε_{t+1} usually does not obey mathematically analytic distribution like Gaussian distribution, which has been validated in our experiments.

3. It is trained by a designed hybrid loss function. We experimentally display that it can achieve better generalization accuracy and variance by training with that loss function. Moreover, it only utilizes back-propagation for training, which makes it friendly implemented by popular deep learning libraries, such as TensorFlow and PyTorch.

DeepPIPE is the first methodology that can overcome all the limitations existing in previous studies, including DeepHQ (Pearce et al., 2018), DeepMVE (Wang et al., 2019a), LUBE (Khosravi et al., 2010), BDL (Fortunato et al., 2017), MC-Dropout (Gal and Ghahramani, 2016), and DE-RNN (Yeo et al., 2018). Its key characteristics are summarized in Table 5.1. It is also flexible and can be enhanced by being combined with popular modules such as attention mechanisms and residual connections.

Table 5.1 Summarized key differences between DeepPIPE and previous methods. PE and PI represents point estimation and prediction interval, respectively.

V.S.		**DeepPIPE**
DeepHQ	PI	PI & PE
DeepMVE	distribution assumption	distribution free
LUBE	simulated annealing	back propagation
BDL	Bayesian inference	non-Bayesian method
MC-Dropout	Monte Carlo	non-Monte Carlo
DE-RNN	need discretization	non-discretization

The rest of this chapter is organized as follows. In Section II, we discuss the related works; in Section III, we introduce the proposed methodology DeepPIPE; in Section IV, we show the experimental results on three real-world datasets; and finally, some conclusions and future works are given in Section V.

5.2 Proposed Methodology

5.2.1 Model framework

DeepPIPE is based on Encoder-Decoder (a.k.a sequence-to-sequence) (Xiao et al., 2018). Both encoder and decoder consist of LSTM which can address gradient

vanishing problem compared with vanilla RNN (Wang et al., 2017c). The formula of the LSTM cell is:

$$\begin{aligned} i_t &= \sigma\left(\mathbf{x}_t U^i + h_{t-1} W^i\right) \\ f_t &= \sigma\left(\mathbf{x}_t U^f + h_{t-1} W^f\right) \\ o_t &= \sigma\left(\mathbf{x}_t U^o + h_{t-1} W^o\right) \\ \tilde{C}_t &= \tanh\left(\mathbf{x}_t U^z + h_{t-1} W^z\right) \\ C_t &= \sigma\left(f_t \odot C_{t-1} + i_t \odot \tilde{C}_t\right) \\ h_t &= tanh(C_t) \odot o_t \end{aligned} \tag{5.6}$$

where

- $\mathbf{x}_t$ defines the input vectors at the timestep t,
- i_t defines the input gates, f_t defines the forget gates and o_t defines the output gates,
- $\tilde{C}_t$ defines the temporary hidden state which is computed based on the current input and the previous hidden state. C_t defines the internal memory.
- h_t defines the output hidden states, computed by multiplying the memory with the output gate. h_{t-1} defines the hidden states at the previous time step,
- W and U define the learnable weights,
- σ and $tanh$ define sigmoid and tanh activation function respectively,
- $\odot$ defines the element-wise product.

The encoder $E(\cdot)$ first encodes input series $\mathbf{X}_{1:t}$ to context vectors $\mathbf{c}$:

$$\mathbf{c} = E(\mathbf{X}_{1:t}; \theta_1) \tag{5.7}$$

$\mathbf{c}$ is then transferred to form the initial state of decoder $D(\cdot)$ which decodes $\mathbf{c}$ and consequently generates the sequential prediction interval and point estimation simultaneously:

$$\hat{\mathbf{y}}_{t+1:t+S}, \hat{\mathbf{y}}^{L}_{t+1:t+S}, \hat{\mathbf{y}}^{U}_{t+1:t+S} = D(\mathbf{c}; \theta_2) \tag{5.8}$$

where θ_1 and θ_2 are learnable parameters. For each time step $t+s$, $\hat{y}_{t+s}, \hat{y}^{L}_{t+s}$ $y_{t+s}$and $\hat{y}^{U}_{t+s}$ are calculated through below transformation:

$$\begin{aligned} \hat{y}_{t+s} &= h_{t+s}U^{p} + B^{p} \\ \hat{y}^{L}_{t+s} &= h_{t+s}U^{l} + B^{l} \\ \hat{y}^{U}_{t+s} &= h_{t+s}U^{u} + B^{u} \end{aligned} \tag{5.9}$$

where U and B are learnable parameters shared among total time steps. The architecture of DeepPIPE is illustrated in Figure 5.1, where the *Assumptions* and *Hybrid loss function* are explained in the next section.

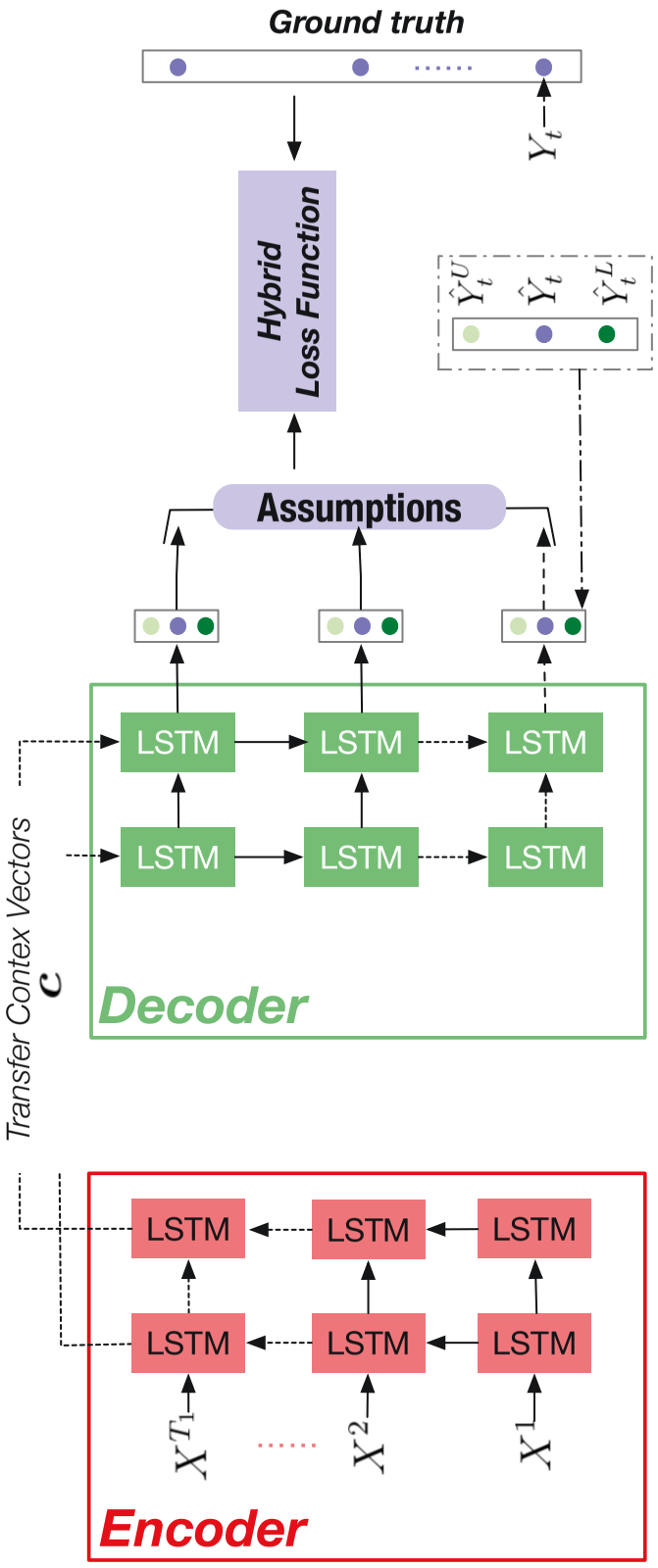


Figure 5.1 Framework of DeepPIPE. The encoder $E(\cdot)$ first encodes input series $\mathbf{X}_{1:t}$ to hidden states $\mathbf{c}$ which is transferred to form the initial state of decoder $D(\cdot)$. $D(\cdot)$ then decodes $\mathbf{c}$ and consequently generates the sequential prediction interval and point estimation simultaneously. The proposed hybrid loss function based on related assumptions is adopted to train the whole network.

5.2.2 Hybrid loss function

In order to equip DeepPIPE with the capability of point estimation and prediction interval, the training loss function L consists of two parts, where L_{PE} is to penalize the loss of point estimation and L_{PI} is to penalize the loss of prediction interval. The hyper-parameter β balances the influence between point estimation and prediction interval. Formally, it is defined as below:

$$L = \beta L_{PE} + L_{PI} \tag{5.10}$$

Particularly, L_{PE} is denoted as below:

$$L_{PE} = \frac{1}{NS} \sum_{n=1}^{N} \sum_{s=1}^{S} v_s \cdot |y_{t+s}^{(n)} - \hat{y}_{t+s}^{(n)}| \tag{5.11}$$

where N is the training batch size, S is the number of forecasted time steps, v_s is a weight factor for the time step s, $y_{t+s}^{(n)}$ and $\hat{y}_{t+s}^{(n)}$ are the ground truth and our prediction at the timestep $t+s$, respectively.

To learn prediction interval, L_{PI} is inspired from the study (Pearce et al., 2018) and constructed as below:

$$L_{PI} = \lambda \frac{N}{(1-P_c)P_c} max(0, (P_c - PICP))^2 + MPIW \tag{5.12}$$

The advanced intuitions of the formula (5.12) is, during training process, if PICP is lower than the expected confidence level P_c, a penalty loss will be imposed through the operation $max(0,\ (P_c - PICP))^2$. The width of the prediction interval is meanwhile accomplished as small as possible by plus MPIW. N corresponds to the training batch size. The expression $\lambda \frac{N}{(1-P_c)P_c}$ is the derivative which can be

referred to the original paper (Pearce et al., 2018) for the details. In particular, λ is a weighting hyper-parameter that controls the loss balance between PICP and MPIW. To address multi-step forecasting in time series scenarios, PICP is denoted as:

$$PICP = \frac{c}{N \cdot S} \tag{5.13}$$

where $c = \sum_{n=1}^{N} \sum_{s=1}^{S} b_{t+s}^{(n)}$ is the total number of *captured data points*. Specially, we say a forecasting value $y_{t+s}^{(n)}$ (value of the nth training sample at the time step $t+s$) captured, shall satisfy:

$$b_{t+s}^{(n)} = \begin{cases} 1, & \text{if} \quad \hat{y}_{t+s}^{(n),L} \leq y_{t+s}^{(n)} \leq \hat{y}_{t+s}^{(n),U} \\ 0, & \text{else.} \end{cases} \tag{5.14}$$

where $\hat{y}_{t+s}^{(n),L}$ and $\hat{y}_{t+s}^{(n),U}$ are the forecasted lower and upper bound for the the nth training sample at the time step $t+s$. MPIW is defined as:

$$MPIW = \frac{1}{c} \sum_{n=1}^{N} \sum_{s=1}^{S} w_s \cdot (\hat{y}_{t+s}^{(n),U} - \hat{y}_{t+s}^{(n),L}) \cdot b_{t+s}^{(n)} \tag{5.15}$$

where w_s is a weight factor for the time step s. Note that $\frac{1}{c}$ indicates that only the captured forecasting values are considered for the calculation of MPIW. Due to the existence of operation $max(\cdot)$ in (5.12), applying the Boolean $b_{t+s}^{(n)}$ in formulas (5.13) and (5.15) can cause the L_{PI} not to be differentiable for back-propagation algorithm. To this end, $b_{t+s}^{(n)}$ is substituted with a softened version as below:

$$b_{t+s}^{(n)} = \sigma(s \cdot (\hat{y}_{t+s}^{(n),U} - y_{t+s}^{(n)})) \cdot \sigma(s \cdot (y_{t+s}^{(n)} - \hat{y}_{t+s}^{(n),L})) \tag{5.16}$$

where σ is the sigmoid activation function and s is a hyper-parameter factor for softing. Accordingly, the softened $b_{t+s}^{(n)}$ is adopted in formulas (5.13) and (5.15).

Explanation of assumptions The formula (5.12) in the original study (Pearce et al., 2018) does not consider the time dimension; hence, it can not be applied directly for modeling time series. To address this problem, we expand it to be compatible with the sequence-to-sequence model based on assumptions as follows.

- Different time steps $t+s$ have an equal weight factor as one. That is to say, in formula (5.11) and (5.15), in this research, v_s and w_s are set to 1.
- Although this research only considers the single variable forecasting, when DeepPIPE is adopted for multiple variables forecasting, the weight factors of different variables can be set according to users' experience.

5.2.3 Training algorithm

Algorithm 1 outlines the DeepPIPE training procedure. We first construct the data pairs from historical time-series data. A sliding window with sliding step one is applied to the raw time series. For the timestep i, we pair $\mathbf{X}_{1:t}$ and $\mathbf{y}_{t+1:t+S}$ as a training sample, which is shown by lines 1-6. The model is then trained through batch gradient descending and back-propagation algorithm to minimize the loss function L, which is illustrated by lines 7-11.

5.3 Experiments and Analysis

In this section, we first introduce the datasets, experimental settings, evaluation metrics, evaluation metrics, and baselines. Afterward, we conduct a performance analysis for the experimental results.

Algorithm 5.1: DeepPIPE Training Algorithm

Input : Historical input time series: $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_I$;
Historical target time series: $y_1, y_2, ..., y_I$;
Input window size t;
Output window size S;
Confidence level P_c (0.9 in our experiments).

Output : Learned DeepPIPE model

// Format the training dataset by sliding window

$\mathcal{D} \leftarrow \emptyset$

while *all available timestamp i* $(0 \leq i \leq I - t - S)$ **do**

  $\mathbf{X}_{1:t} = [\mathbf{x}_{1+i}, \mathbf{x}_{2+i}, ..., \mathbf{x}_{t+i}]$

  $\mathbf{y}_{t+1:t+S} = [y_{t+1+i}, y_{t+2+i}, ..., y_{t+S+i}]$

  put a data sample $(\mathbf{X}_{1:t}, \mathbf{y}_{t+1:t+S})$ into $\mathcal{D}$

end

// Split dataset $\mathcal{D}$ into training set $\mathcal{D}_1$, validation set $\mathcal{D}_2$, and test set $\mathcal{D}_3$.

// Train

Initialize all trainable parameters θ_1, θ_2 in DeepPIPE.

repeat

  randomly select a batch of samples $\mathcal{B}$ from $\mathcal{D}_1$

  update parameters θ_1, θ_2 by minimizing the loss function with $\mathcal{B}$

until *stopping criteria are met*;

5.3.1 Datasets

Three open real-world datasets *appliances energy* and *air quality* are used in our experiments. Figure 5.2 displays examples of three time-series datasets. The detail of each dataset is introduced as follows.

Multivariate datasets Appliances energy UCI dataset was collected for 4.5 months with 10-min frequency, having a total 19737 records (Candanedo et al., 2017)[1]. One-hot encoding is utilized to transform the category variables (e.g., DayOfWeek and TimeOfDay), and the final feature dimensions are 36 for each time step, and the dimension *appliances energy consumption* is the target variable.

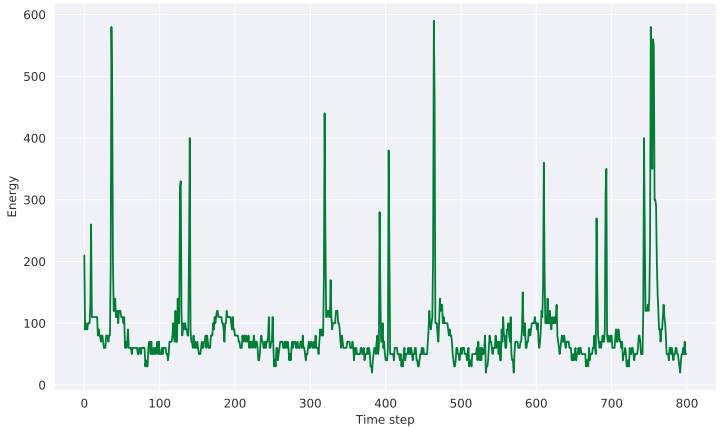
Air quality dataset was collected for the air quality forecasting research (Yi et al., 2018)[2]. This dataset is collected from 00:00 01/May/2014 to 22:00 30/Apr/2015 at 1-hour frequency, a totally of 8759 records. Unfortunately, there are serious missing value problems in the majority of air monitoring stations. We finally select the station *Yangjiang* as our data source, which has the least missing values. We have six pollutants including $PM_{2.5}$, PM_{10}, NO_2, CO, O_3, and SO_2 to build previous feature series and take the future $PM_{2.5}$ as the target variable.

Univariate dataset Brent oil price dataset contains daily Brent oil prices from 17th of May 1987 until the 30th of September 2019 at 1-day frequency [3], totally 8216 records.

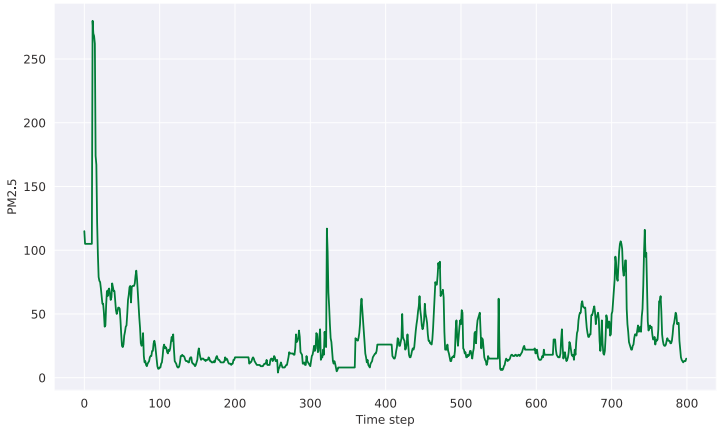
[1]Data link: https://archive.ics.uci.edu/ml/datasets/Appliances+energy +prediction. We investigated that there was a technical misuse in the original research paper (Candanedo et al., 2017). In the source codes, the authors shuffled training/testing data partitioning, which was a misuse of time-series data. We split the data in chronological order without shuffle operation. Hence the MAE in our experimental results and in the original paper are not comparable because of forecasting implemented on the different test datasets.

[2]Data link: http://urban-computing.com/data/Data-1.zip

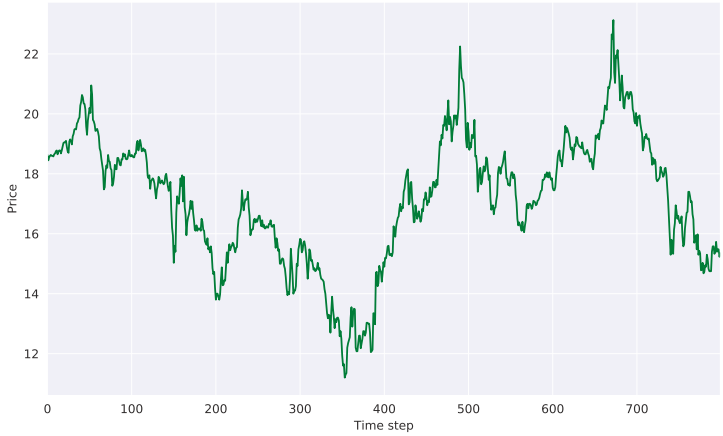
[3]Data link: https://www.kaggle.com/mabusalah/brent-oil-prices



(a) Clear cycle in appliances energy dataset



(b) Numerous abrupt points in air quality dataset



(c) Complicated dynamics in oil price dataset

Figure 5.2 Examples of three time-series datasets. Each dataset illustrates the first 800-step time series.

5.3.2 Experimental settings and baselines

Data preprocessing We use min-max normalization to normalize the continuous features into [0, 1] and utilize linear interpolation for missing data imputation. In the evaluation, we rescale the predicted values back to the normal scale.

Hyper-parameter settings For a fair comparison, all deep learning methods are configured with the same hyper-parameter as follows. In particular, we set $\lambda = 2$, $\beta = 1.5$, $s = 160$ and $P_c = 0.9$ in formula (5.12) and (5.16). We set the hidden nodes of each layer as 64. We set the batch size as 256. We set the time window size of input feature series as $t = 12, 6, 7$ for datasets appliances energy, air quality, and Brent oil price respectively. For the single forecasting, we set $S = 1$, which indicates to predict the next value once time. For the multi-step forecasting, we set $S = 3$, which means to forecast the following three values once time. Financial time-series prediction is a challenging task, and long-term forecasting is often unhelpful, hence 1-step-ahead setting has become the first consideration (Niaki and Hoseinzade, 2013). Our experiments follow the 1-step-ahead setting for the oil price prediction. All datasets are formatted according to the Algorithm 5.1. The resultant dataset for training, validation, and test are outlined in Table 5.2.

Optimization method Adam (Kingma and Ba, 2014) is adopted as the optimization method to learn the parameters with learning rate 0.001, and early-stopping is implemented on the validation set to avoid overfitting.

Experimental environment All models are trained on a server with Quadro P4000 GPU. The programming environment is Python 3.6 and Tensorflow.

Baselines As mentioned that LUBE (Khosravi et al., 2011c) requires heuristic algorithms and BDL (Fortunato et al., 2017) requires variational inference for optimization. Both two methods not only cannot be seamlessly integrated with

Table 5.2 Information of three real-world datasets. 1/3-step prediction indicate that datasets are formatted for *n*-step-ahead prediction.

Dataset	Feature dimensions	Size training	validation	test
1-step prediction				
Brent Oil Price	1	5743	1228	1229
Appliances Energy	36	13312	1480	4931
Air Quality	6	7089	788	876
3-step prediction				
Appliances Energy	36	13311	1479	4931
Air Quality	6	7087	788	876

popular gradient-based frameworks, such as Tensorflow. Although MC-Dropout (Zhu and Laptev, 2017) only needs back-propagation during training, it requires Monte Carlo by repeating forward calculation for thousands of times to inference prediction interval and point estimation. Moreover, the above methods are time-consuming and do not show significant improvement compared to the MVE method according to previous researches (Khosravi et al., 2011b; Lakshminarayanan et al., 2017). In this respect of pragmatism, their shortcomings have already made them lose their advantage compared with DeepPIPE, and thus, we compare DeepPIPE with below baselines.

- **DeepMVE** adopts the Encoder-Decoder architecture extended from MVE model (Wang et al., 2019a). Although MVE loss can be easily integrated into deep learning for uncertainty quantification, a strong assumption is that the ground truth $y_{t+s} \sim N(\hat{y}_{t+s}, \sigma_{t+s})$, and the interval width can be calculated by looking up the z-score table of Gaussian distribution according to the pre-defined P_c. In our experiments, IW_{t+s} is calculated by $2 * 1.64 * \sigma_{t+s}$ because

of P_c set to 0.9. The learning process is essentially by maximum likelihood estimation. It has the same hyper-parameter settings with DeepPIPE.

- **DeepHQ** has the same hyper-parameter settings with DeepPIPE except that its loss function only includes prediction interval part L_{PI}. It can also be regarded as a sequence-to-sequence extension from the previous study (Pearce et al., 2018). This baseline aims to illustrate the effectiveness of the designed hybrid loss function, which optimizes L_{PE} and L_{PI} simultaneously.

- **ShallowPIPE** has the same hyper-parameter settings with DeepPIPE except only having one hidden layer. This baseline aims to show the influence of the layer number when compared with DeepPIPE.

Since weights initialization can influence the performance of deep models. In our experiments, each deep model was trained up to five times with different random seeds. The test results are reported with the expression as *mean* $\pm$ *standard deviation*. In addition, non-deep learning baselines include:

- **ARIMA** (Autoregressive Integrated Moving Average) is a popular method for univariate time series forecasting (Benvenuto et al., 2020). It was implemented by *statsmodels.tsa.arima_model.ARIMA*.

- **VAR** (Vector Auto Regression) is one of the most commonly used parametric methods for time series forecasting (Haslbeck et al., 2021). It was implemented by *statsmodels.tsa.api.VAR*.

- **SVR** (Support Vector Regression) is a category of SVM for regression tasks (Chen et al., 2017). It was implemented by the Python package *sklearn.svm.SVR*.

- **GBDT** (Gradient Boosting Decision Tree) is a popular boosting method that ensembles a set of weak decision trees in order (Persson et al., 2017). It was implemented by the Python package *sklearn.ensemble.GradientBoostingRegressor*.

5.3.3 Experimental evaluations

Measurement of point estimation During test, MAE is calculated to evaluate the predictive error of point estimation as below:

$$MAE = \frac{1}{NS}\sum_{n=1}^{N}\sum_{s=1}^{S}|y_{t+s}^{(n)} - \hat{y}_{t+s}^{(n)}| \tag{5.17}$$

where $y_{t+s}^{(n)}$ and $\hat{y}_{t+s}^{(n)}$ are the ground truth and the prediction for the test sample n at the timestep $t+s$, respectively.

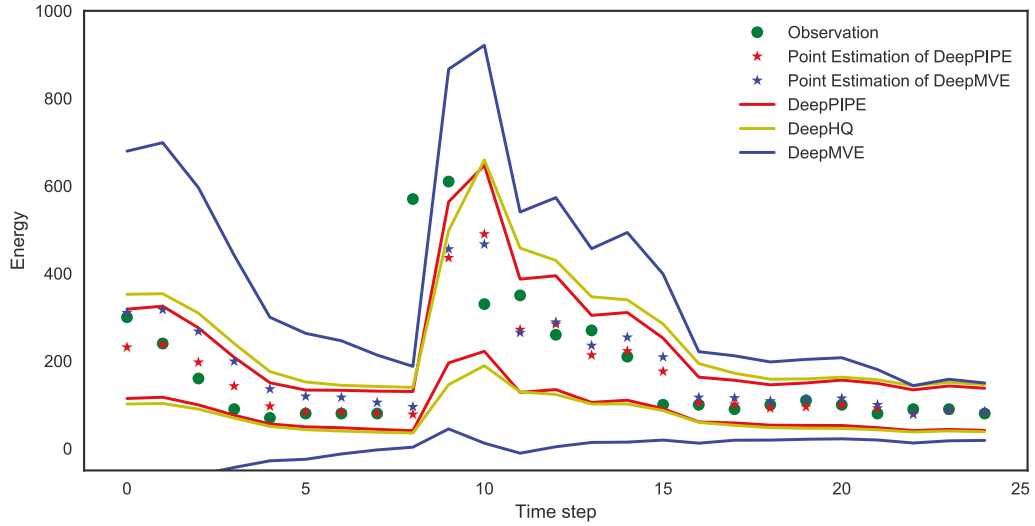
Measurement of uncertainty quantification PICP in formula (5.13) and MPIW in formula (5.15) are taken as the measurements of uncertainty quantification.

5.3.4 Performance analysis

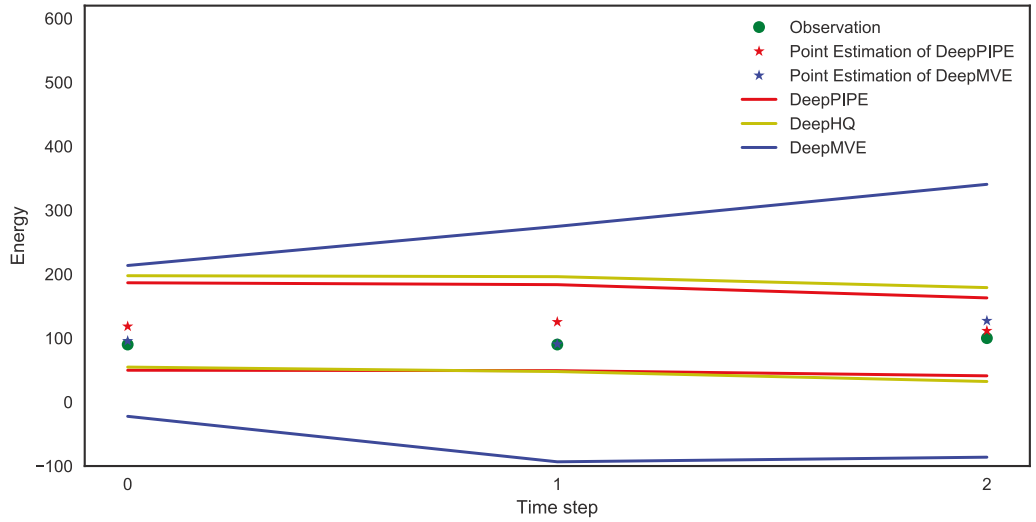
Table 5.3, 5.4 and 5.5 report the experimental results. Figure5.3 and Figure5.4 exhibit the visualization of forecasting examples. We conduct analysis from aspects as follows.

Performance of point estimation

Our top-line conclusions are that DeepPIPE almost achieved the best MAE measurement on three datasets for both 1-step and 3-step forecasting. We can see that on appliances energy dataset, it achieved 34.99±0.61 for 1-step prediction and 41.77±1.03 for 3-step prediction. For the air quality dataset, it achieved



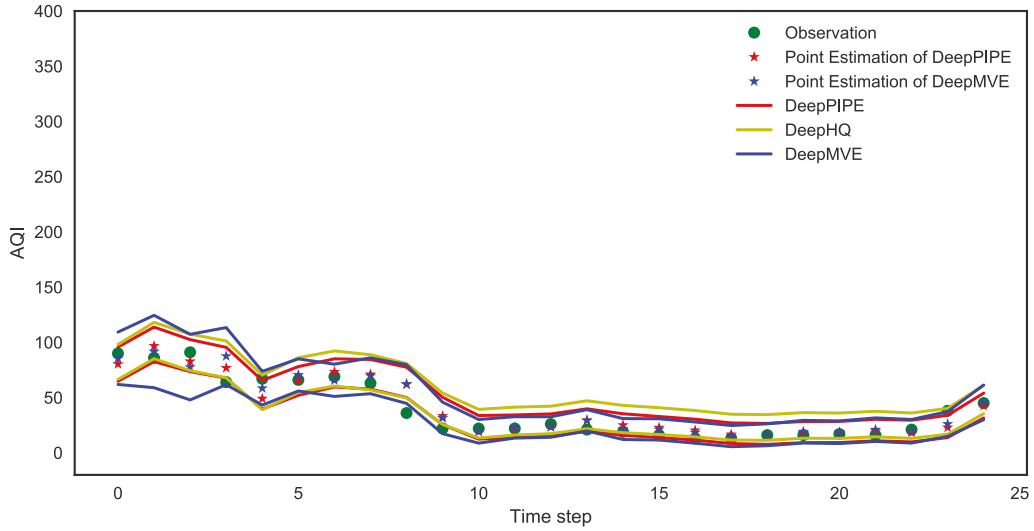
(a) 25 test examples of 1-step-ahead forecasting.



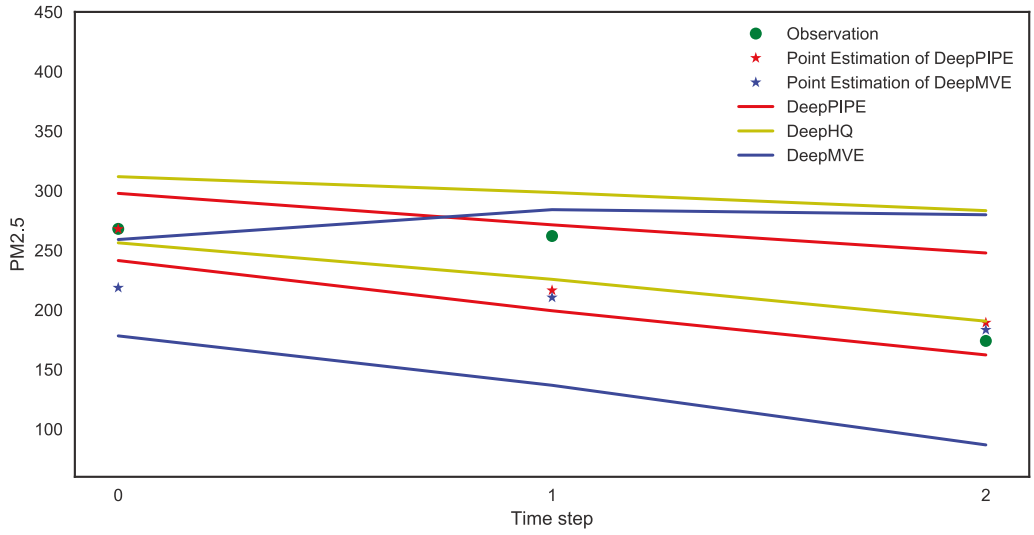
(b) One example of end-to-end 3-step-ahead forecasting

Figure 5.3 Predictive examples of DeepPIPE on appliances energy dataset. DeepPIPE achieved better MPIW significantly than DeepMVE.

5.25±0.09 for 1-step prediction and 7.69±0.03 for a 3-step prediction. DeepHQ has no ability for point estimation, so the relevant results were left out. For the 1-step forecasting of Brent oil price, ARIMA achieved the best 1.14±0.00, whereas DeepPIPE attained 1.22±0.17. The result is not impossible because the financial



(a) 25 test examples of 1-step-ahead forecasting.



(b) One example of end-to-end 3-step-ahead forecasting

Figure 5.4 Forecasting examples of DeepPIPE on air quality dataset. DeepPIPE achieved better MPIW significantly than DeepMVE.

dynamic is highly complicated, and the classic baseline ARIMA can sometimes be strong and perform better. It is noticeable that DeepPIPE has obvious advantages when modeling prediction interval, which is analyzed as below.

Performance of prediction interval

All methods have achieved the predefined confidence level of P_c, which indicated that all PICPs were greater than 0.9. However, under this premise, the performance of MPIW had obvious distinctions. Methods like ARIMA, VAR, GBDT, DeepMVE significantly overestimated MPIW. Our explanation for this was that these models assumed that the ground truth obeyed Gaussian distribution, which might be far different from the realistic data distribution. The observations that are consistent with our explanation are that DeepHQ, ShallowPIPE, and DeepPIPE, which all are distribution-free, accomplished smaller MPIW. Notably, DeepPIPE accomplished the best MPIW performance 106.96±2.39 (1-step) and 116.52±1.69 (3-step) on appliances energy dataset, 22.79±0.47 (1-step) and 29.95±0.12 (3-step) on air quality dataset and 3.26±0.28 on Brent oil price dataset. The results of SVR were omitted, since *sklearn.svm.SVR* has not provided the function for the prediction interval.

Performance of stability

Deep models trained with various initialized parameters can lead to different forecasting results. To address the randomness problem, we implemented each deep model with different random seeds up to five times and calculated the standard deviation expressed via $\pm X$ to reflect the forecasting stability. In our experiments, DeepPIPE achieved the least standard deviation, and it illustrated that DeepPIPE had a stable generalization. Although methods including ARIMA, VAR, SVR, and GBDT, were configured by certain hyper-parameters leading to deterministic solutions, thus the resultant standard deviations equal to zeros, they usually have poor performances on MAE and MPIW.

Effect of hybrid loss function On appliances energy dataset for 1-step forecasting, DeepHQ achieved MPIW as 115.75±5.71 while DeepPIPE achieved

smaller MPIW as 106.96±2.39. For 3-step forecasting, DeepHQ achieved MPIW as 119.20±3.82 while DeepPIPE achieved smaller MPIW as 116.52±1.69. For the air quality dataset for 1-step forecasting, DeepHQ achieved MPIW as 23.71±0.61 while DeepPIPE achieved smaller MPIW as 22.79±0.47. For 3-step forecasting, DeepHQ achieved MPIW as 31.02±0.57 while DeepPIPE achieved smaller MPIW as 29.95±0.12. On the Brent oil price dataset, DeepHQ achieved MPIW as 3.72±0.12 while DeepPIPE achieved smaller MPIW as 3.26±0.28. These observations indicated that combining L_{PI} and L_{PE} rather only L_{PE} as loss function indeed improved the performance of prediction interval. Our explanation for this was that it benefited from the multi-task learning, which can promote consistency and coherence for point estimation and prediction interval.

Effect of deep learning By comparing DeepPIPE with ShallowPIPE, we can see that stacked two layers helped DeepPIPE improve its performance for both point estimation and prediction interval on three datasets. For example, on appliances energy dataset, ShallowPIPE achieved MAE as 36.31±0.85, MPIW as 113.54±6.70, while DeepPIPE achieved smaller MAE as 34.99±0.61, MPIW as 106.96±2.39. The experimental results indicated the proposed DeepPIPE was compatible with the philosophy of deep learning. We believe its performance can be improved further when combined with other advanced deep learning techniques, such as attention mechanism and memory network.

The advantage of distribution-free assumption As shown in Figure 5.3 and Figure 5.4, a noteworthy observation was that DeepMVE forecasted the point estimation located at the central location of the prediction interval, which was the consistent result of the Gaussian assumption. However, the point estimation of DeepPIPE does not necessarily locate at the central location between the prediction

Table 5.3 Experimental results on Brent oil price dataset. DeepPIPE achieved the best on MAE and MPIW.

Model	MAE	PICP	MPIW
1-step prediction			
ARIMA	1.14±0.00	1.00±0.00	3.45±0.00
SVR	1.23±0.00	-	-
GBDT	1.38±0.00	0.98±0.00	4.54±0.00
DeepMVE	1.21±0.28	0.96±0.02	4.04±0.40
DeepHQ	-	0.99±0.03	3.42±0.12
ShallowPIPE	1.18±0.21	0.99±0.01	3.38±0.33
DeepPIPE	**1.02±0.17**	1.00±0.00	**3.26±0.28**

Table 5.4 Experimental results on appliances energy dataset. DeepPIPE achieved the best on MAE and MPIW.

Model	MAE	PICP	MPIW
1-step prediction			
ARIMA	36.17±0.00	1.00±0.00	233.05±0.00
VAR	36.91±0.00	1.00±0.00	232.21±0.00
SVR	41.06±0.00	-	-
GBDT	36.70±0.00	1.00±0.00	173.84±0.00
DeepMVE	38.52±2.35	0.95±0.02	152.33±5.61
DeepHQ	-	0.93±0.00	115.75±5.71
ShallowPIPE	36.31±0.85	0.93±0.01	113.54±6.70
DeepPIPE	**34.99±0.61**	0.93±0.00	**106.96±2.39**
3-step prediction			
ARIMA	42.68±0.00	1.00±0.00	279.85±0.00
VAR	44.23±0.00	1.00±0.00	275.39±0.00
SVR	44.67±0.00	-	-
GBDT	43.51±0.00	1.00±0.00	204.23±0.00
DeepMVE	45.53±3.21	0.95±0.02	141.97±18.06
DeepHQ	-	0.93±0.00	119.20±3.82
ShallowPIPE	48.17±1.71	0.93±0.01	123.67±2.52
DeepPIPE	**41.77±1.03**	0.93±0.01	**116.52±1.69**

interval. This observation proves the superiority of DeepPIPE to model sensible distribution from the observed data.

Table 5.5 Experimental results on air quality dataset. DeepPIPE achieved the best on MAE and MPIW.

Model	MAE	PICP	MPIW
1-step prediction			
ARIMA	5.35±0.00	1.00±0.00	26.30±0.00
VAR	5.40±0.00	1.00±0.00	27.10±0.00
SVR	5.50±0.00	-	-
GBDT	5.49±0.00	-	25.89±0.00
DeepMVE	5.47±0.29	0.93±0.00	23.69±0.79
DeepHQ	-	0.93±0.00	23.71±0.61
ShallowPIPE	5.65±0.34	0.93±0.00	23.62±0.66
DeepPIPE	**5.25±0.09**	0.93±0.01	**22.79±0.47**
3-step prediction			
ARIMA	7.85±0.00	1.00±0.00	37.46±0.00
VAR	7.94±0.00	1.00±0.00	36.01±0.00
SVR	8.00±0.00	-	-
GBDT	8.03±0.00	-	34.52±0.00
DeepMVE	7.80±0.24	0.93±0.01	32.31±0.82
DeepHQ	-	0.91±0.00	31.02±0.57
ShallowPIPE	8.01±0.27	0.91±0.00	30.42±0.58
DeepPIPE	**7.69±0.03**	0.91±0.00	**29.95±0.12**

5.4 Summary

In this chapter, we proposed a deep learning model called DeepPIPE for time series forecasting. A novel loss function has been designed to cast the problems as multi-task learning, thus making DeepPIPE be capable of implementing prediction interval and point estimation simultaneously. We have qualitatively analyzed its superiority over previous methods. To summarize, DeepPIPE can be trained by back-propagation algorithm, which can be seamlessly deployed in popular deep learning libraries and embrace admirable hallmarks compared with LUBE, BDL, and MC-Dropout, of which they required heuristic algorithms, variational inference, and

Monte Carlo, respectively. Furthermore, different from DeepMVE, DeepPIPE can automatically learn the distribution of ε_{t+s} without prior distribution assumptions. It also overcome the drawbacks of DE-RNN. To the best of our knowledge, this was the first methodology that can overcome all the limitations existing in the previous studies. We also quantitatively shown its considerable performance and forecasting stability on three real-world datasets. For the future works, we will explore DeepPIPE in real-world applications, such as energy forecasting (Hong et al., 2019) and electrocardiogram analysis (Perc, 2005). We also aim to incorporate advanced techniques, including attention mechanisms and memory networks, to further improve its performance.

Chapter 6

Deep Forecasting via Quantile Fusion

In Chapter 3, we proposed a deep point estimation method for regional traffic flow prediction. During the research process, we have found that the contribution of the current deep forecasting methods are mainly concentrated on the design of the network structure, and the predictive target are mostly focused on point estimation. However, due to the different characteristics of different task datasets, the complex deep model structure specially designed for a prediction task is difficult to be widely applied to other prediction tasks. It is necessary to measure the uncertainty of the forecast results. Chapters 4 and 5 of this thesis proposed a widely applicable deep uncertainty quantification from the perspective of loss function design. This chapter further proposes a deep quantile fusion method, which improves the generalization and robustness of point estimates by means of an uncertainty quantification method, quantile regression. Experiments are validated and evaluated on eight UCI regression datasets and one time series dataset. The experimental results has also demonstrated the effectiveness of the proposed method.

6.1 Introduction

Forecasting techniques are widely employed to solve regression tasks where the goal is to predict continuous values. Over recent years, deep learning has been revolutionizing this field. Exploring deep forecasting has achieved great success in real-world applications, such as rainfall prediction (Qiu et al., 2017), car-calling demand forecasting (Geng et al., 2019; Yao et al., 2018), healthcare data modeling (Gao et al., 2019; Zhang et al., 2019), mobility estimation (Wang et al., 2018b), air quality prediction (Yi et al., 2018), recommender system (Lu et al., 2018), trust prediction (Wang et al., 2019b), etc. Although much endeavor has been devoted to developing a deep architecture to achieve state-of-the-art performance on one specific task, how to adapt them to other tasks is not an easy mission [1]. For instance, a three-way CNN model is proposed for crowd flow prediction (Zhang et al., 2018). While its architecture is non-trivial to be extended for modeling data without conspicuous cycles and grid-style input, it indicates that a generic deep forecasting methodology is more scalable than a specific architecture.

Generic deep learning methodologies can be constructed from the following perspectives (Ian Goodfellow and Courville, 2016): 1) Basic architectures, such as Seq2Seq, Graph Neural Networks, etc. 2) Effective modules, such as attention mechanism, normalization layer, etc. 3) Optimization algorithms, such as Adam, RMSProp, etc. 4) Loss functions, such as MAE/MSE in regression and cross-entropy loss in classification. Among these aspects, designing novel architectures and modules becomes pervasive. Nevertheless, a recent keynote (Rudin, 2019) highlighted a philosophy that *it is not clear that extra and complex structure*

[1] This study only considers scenarios where the forecasting model is developed on one dataset, so it is not related to transfer learning.

is always necessary or helpful. Models can often be made easier by adding interpretability constraints, which shrink the hypothesis space. The less noticeable aspect is the usage of loss functions. A recent study of weather forecasting (Wang et al., 2019a) reported a novel phenomenon that training the vanilla Seq2Seq model with Gaussian-based log-likelihood loss function instead of MSE can significantly improve the model generalization. Quantile regression (QR) has been proposed by designing the quantile loss, a.k.a. pinball loss, to estimate conditional quantiles to model distribution information (Koenker and Bassett Jr, 1978). With the renaissance of deep learning, a combination of QR and deep neural networks has attracted considerable attention (Rodrigues and Pereira, 2020; Yan et al., 2018; Zhang et al., 2019). iQRNN (Zhang et al., 2019) incorporates quantile loss in deep neural networks to generate multiple quantile forecasts for load forecasting. DeepJMQR (Rodrigues and Pereira, 2020) was first proposed to train deep learning models by combining quantile loss and MSE together, which can play a regularization role in forecasting accuracy and reduce quantile crossing. All these studies demonstrate that the capability of deep forecasting models may not only be related to its complex architecture but also be related to how to utilize loss functions wisely.

In this research, we take inspiration from the above studies and introduce a generic methodology. We propose that when applying deep learning for forecasting, to preliminarily carry out QR to represent target distribution by quantiles, and further fuse the quantiles as features for the forecasted target. The essence of combining QR with deep learning is to utilize a quantile layer to transform hidden features as semantic features, i.e., quantiles. The significant advantages include: 1) quantile features boosts accuracy and robustness, and the detailed experimental analysis is discussed in Section 4; 2) compared with hidden features, quantile

features can additionally provide distribution information that can be referenced for operators; 3) since the method is achieved by constructing loss function, it is model structure independent, and hence is feasible on various deep learning models. We name this methodology as deep quantile fusion (DQF) and empirically show its remarkable accuracy, stability, and robustness. Lastly, although our experiments are mainly implemented on the non-sequential datasets, it is essential to note that the proposed methodology is not restricted by the structure of deep learning models and can be easily applied to general regression problems such as time-series forecasting and spatio-temporal prediction.

In summary, the contributions of this chapter include:

- We propose a generic methodology called DQF for single value forecasting. It first conducts QR to learn quantiles and then take these quantiles as input to forecast the targeted value. Through this, the forecasting accuracy can be improved significantly.

- We devise crossing loss (CL) function to address the open problem *quantile crossing*. The proposed CL can decrease the ratio of quantile crossing considerably.

- We discuss two learning modes, i.e., end-to-end (E2E) and two-stage (2S), and conclude E2E outperforms 2S when applied in the DQF methodology. It helps researchers perceive which learning modes match DQF much better.

- We rigorously evaluate DQF on eight UCI datasets, and for each dataset, execute 20-fold testing to ensure its significant superiority and conduct a comprehensive analysis from various perspectives. We also demonstrate DQF to combine with AutoEncoder as QF-AE to model time series.

The rest of the chapter is organized as follows. Section II introduces the proposed method. Section III discuss experiments and performance analysis. IV further discusses the model extension based on AutoEncoder to model time series. V summarizes the conclusion and future work.

6.2 Deep Quantile Fusion Methods

The methodology is constructed by combining QR and deep learning. This section first retrospects the preliminary of QR and then presents the proposed DQF method.

6.2.1 Preliminary

QR is proposed to estimate conditional quantiles to model distribution information (Koenker and Bassett Jr, 1978). The oracle quantile q^{τ} of the random variable y is defined as $P(y \leq q^{(\tau)}) = \tau$. Given that $\hat{q}^{(\tau)}$ is denoted as the estimation of q^{τ} and it can be estimated by the machine learning model through minimizing QL loss function $QL_{\tau}(d) = max(\tau d, (\tau - 1)d)$ where $d = y - \hat{q}^{(\tau)}$. In particular, when $\tau = 0.5$, minimizing QL is equal to minimizing MAE, which means the $\hat{q}^{(\tau)}$ is the median estimation.

6.2.2 Model framework

Based on QR, the methodology of DQF consists of two parts, which is illustrated in Figure 6.1. The first part is to deploy a neural network N_1 to incorporate input x and forecast multiple quantiles $\hat{\mathbf{q}}$ defined by users. Hence we name the output layer of N_1 as the *quantile layer*. According to the knowledge of QR, such $\hat{\mathbf{q}}$ can

represent the distribution information of the target variable y given x. The second part is to take the $\hat{\mathbf{q}}$ as the following inputs of another neural network N_2 to output the regressed value $\hat{y}$ of y. MSE and QL are the training loss functions and will be introduced in the following section. Mathematically, the entire network following formulas:

$$\hat{\mathbf{q}} = [\hat{q}^{(\tau_1)}, \hat{q}^{(\tau_2)}, ..., \hat{q}^{(\tau_j)}] = N_1(x) \tag{6.1}$$

$$\hat{y} = N_2(\hat{\mathbf{q}}) \tag{6.2}$$

For illustration, both N_1 and N_2 in this study are configured as forward fully connected networks. For example, the *network 1* can be set with 50 hidden nodes. Its functionality is to extract fifty hidden features from the original input layer for predicting quantiles. The *quantile layer* is set with 11 hidden nodes. It hence outputs 11 quantiles which will be fused by the following *network 2*. The *network 2* is configured as 5 hidden nodes. It aims to extract further five hidden features based on the quantiles to forecast the target $\hat{y}$. More specifications are discussed in Section 4.3.

6.2.3 Loss function

The quantile layer is optimized by the classic QL loss function during training as below:

$$QL_{\tau_j}(d_{i,j}) = max(\tau_j d_{i,j}, (\tau_j - 1)d_{i,j}) \tag{6.3}$$

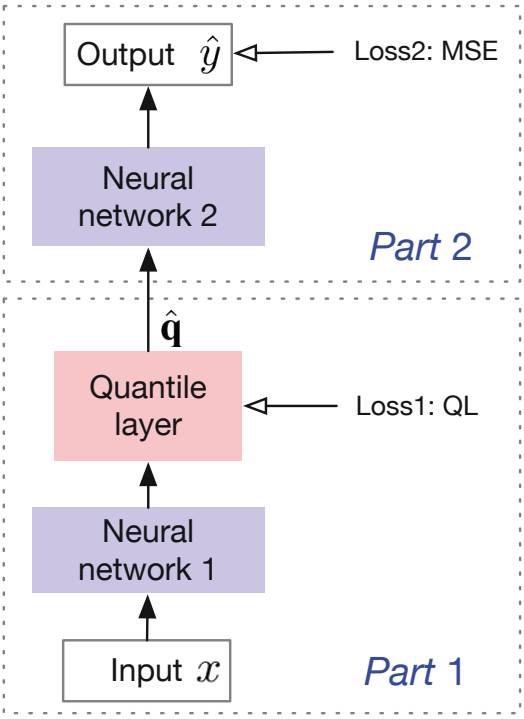


Figure 6.1 The framework of deep quantile fusion.

where τ_j is the jth defined quantile and $d_{i,j} = y_i - \hat{q}_i^{(\tau_j)} \in R$ denotes the residual between the ground truth y_i and the jth forecasted quantile $\hat{q}_i^{(\tau_j)}$ for the data sample i. The total loss for total QR is defined as below:

$$QL = \frac{1}{NQ} \sum_{i=1}^{N} \sum_{j=1}^{Q} QL_{\tau_j}(d_{i,j}) \tag{6.4}$$

where N is the batch size and Q is the number of defined quantiles. An open problem in QR is quantile crossing, which contradicts that the quantile is an increasing function. To address this issue, we propose the below crossing loss.

$$CL = \frac{1}{N(Q-1)} \sum_{i=1}^{N} \sum_{j=1}^{Q-1} \alpha max(0, \beta - (\hat{q}_i^{(\tau_{j+1})} - \hat{q}_i^{(\tau_j)})) \tag{6.5}$$

The hyper-parameter β is set to control the minimum spacing between two quantiles, and α is utilized to penalize crossing for total quantiles.

The final output layer is optimized by MSE loss funtion.

$$MSE = \frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2 \tag{6.6}$$

where $\hat{y}_i$ is the regressed value for the ground truth y_i. The total loss function will consider QL and MSE together:

$$L = QL + MSE \tag{6.7}$$

or if it further considers CL, it follows as below:

$$L = QL + CL + MSE \tag{6.8}$$

6.2.4 End-to-end V.S. two-stage training algorithms

We design two different learning algorithms. The first learning algorithm is conducted in E2E mode. E2E is the most popular learning mode for training deep learning models, which directly optimizes the whole deep model. Algorithm 6.1 outlines its procedure.

A possible problem in the E2E learning mode is that, although E2E is a reasonable choice for the forecasting of y it may not be conducive to the QR. To this end, we hence design another learning algorithm in 2S learning mode, which is shown in Algorithm 6.2.

Algorithm 6.1: E2E learning mode

Input : Input, output and related hyper-parameter.

Output : Learned model.

Initialize all learnable parameters $\theta = [\theta_1, \theta_2]$, where θ_1 and θ_2 are the parameters in network N_1 and network N_2, respectively.

repeat

  Fetch the next training batch $\mathcal{B}$ consisting of batch-size (x_i, y_i) pairs.

  Update θ by the optimizer to minimize the loss function L on $\mathcal{B}$.

until *stopping criteria are met*;

Algorithm 6.2: 2S learning mode

Input : Input, output and related hyper-parameter.

Output : Learned model.

Initialize all learnable parameters $\theta = [\theta_1, \theta_2]$, where θ_1 and θ_2 are the parameters in network N_1 and network N_2, respectively.

repeat

  Fetch the next training batch $\mathcal{B}$.

  Get the output quantiles $\mathcal{Q}$ and update θ_1 by the first optimizer to minimize the loss function QL (or $QL+CL$) on $\mathcal{B}$.

  Fix θ_1 and update θ_2 by the second optimizer to minimize the loss function MSE based on the input quantiles $\mathcal{Q}$.

until *stopping criteria are met*;

6.3 Experiments and Analysis

This section introduces the datasets, experimental environment and hyperparameters, methods and baselines, and result analysis.

6.3.1 Datasets

Our experiments are implemented on eight UCI datasets which were also used for evaluating probabilistic backpropagation (PBP) (Hernández-Lobato and Adams, 2015), MC-dropout (Gal and Ghahramani, 2016) and deep ensemble (Lakshminarayanan et al., 2017). Each dataset is split into 20 train/test folds, except the dataset Protein, which is split into five folds, and the size of train/test data ratio is 9:1. Data is normalized by z-score normalization with the mean and standard deviation of the training set and are de-normalized when evaluated.

6.3.2 Experimental settings and baselines

The experimental environment is as follows: OS of Red Hat Enterprise Linux Workstation (7.5), GPU of Tesla V100, CUDA 10.0 and cuDNN 7.4. We use Pytorch (1.1.0) to code all of our methods and baselines, since Pytorch has a favorable coding flexibility. The training time depends on the performance of the hardware devices and the dataset size. For example, on the largest dataset Protein, it takes around 47 seconds to train a model. Once the training is complete, the online testing of a single sample is less than 0.1 second.

Note that the experimental goal is not to fine-tune a state-of-the-art model on each dataset, but to verify the robustness and effectiveness of DQF in general. Therefore, the hyperparameters for each method and baseline are set as below:

1) It only uses the training set to train the model. It does not split the validation dataset from the training dataset for comprehensive grid search of hyper-parameter.

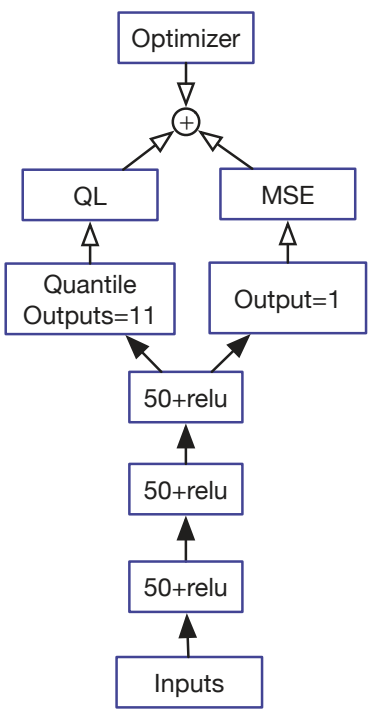
2) For every train/test fold (total 20 folds, except five folds for the dataset Protein) on each UCI dataset, each method or baseline is initialized with the same random seed, trained up to 200 epochs on the training set and then tested on the testing set. There is no early-stopping mechanism.

3) Training batch size is set to 128 on each training set; The optimizers are chosen as Adam, and the learning rate is set to 0.005 for each Adam optimizer.

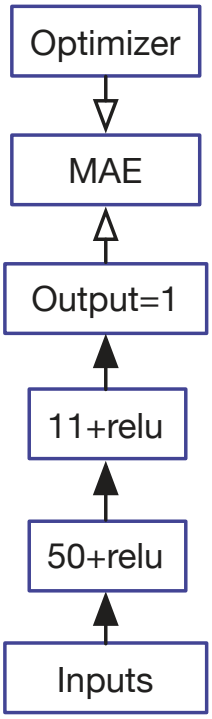
4) The quantile layer is set to have 11 nodes, which aim to forecast 11 usual quantiles and they are 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 95%. We set the penalty weight $\alpha = 1$ and the margin $\beta = 0.01$ in our experiments.

Through these settings, we ensure strictly consistent conditions for all methods and baselines.

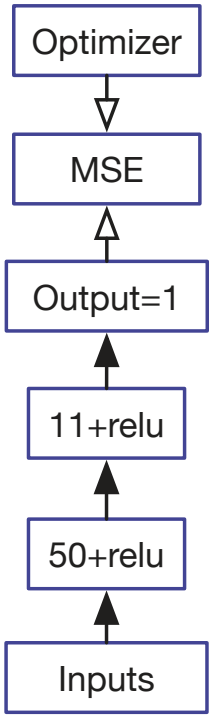
To clearly illustrate the effectiveness of DQF, we divide all methods and baselines into three groups, i.e., DeepJMQR, DNN(50-11)-based models and DNN(50-11-5)-based models. The implication of symbol *DNN(50-11-5)* is that the forward deep neural network (DNN) has three hidden layers with hidden nodes 50, 11 and five respectively, of which the numbers 50 and 5 are two hyper parameters and the number 11 represents the defined 11 quantiles. Compared with DNN(50-11), the superiority of DNN(50-11-5) which owns the last five nodes, will be analyzed later. Particularly, methods belonging to DQF are marked with the suffix ':DQF'.



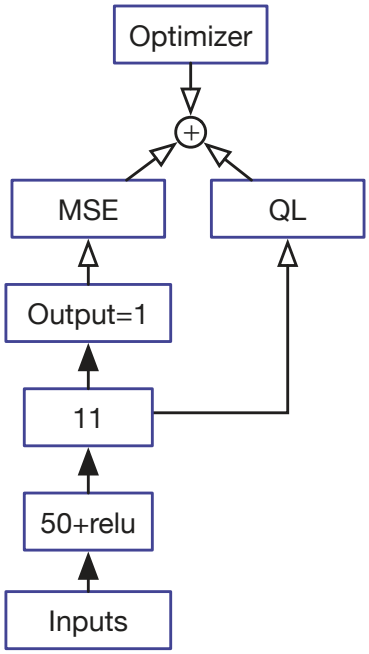
(a) DeepJMQR.



(b) [MAE].

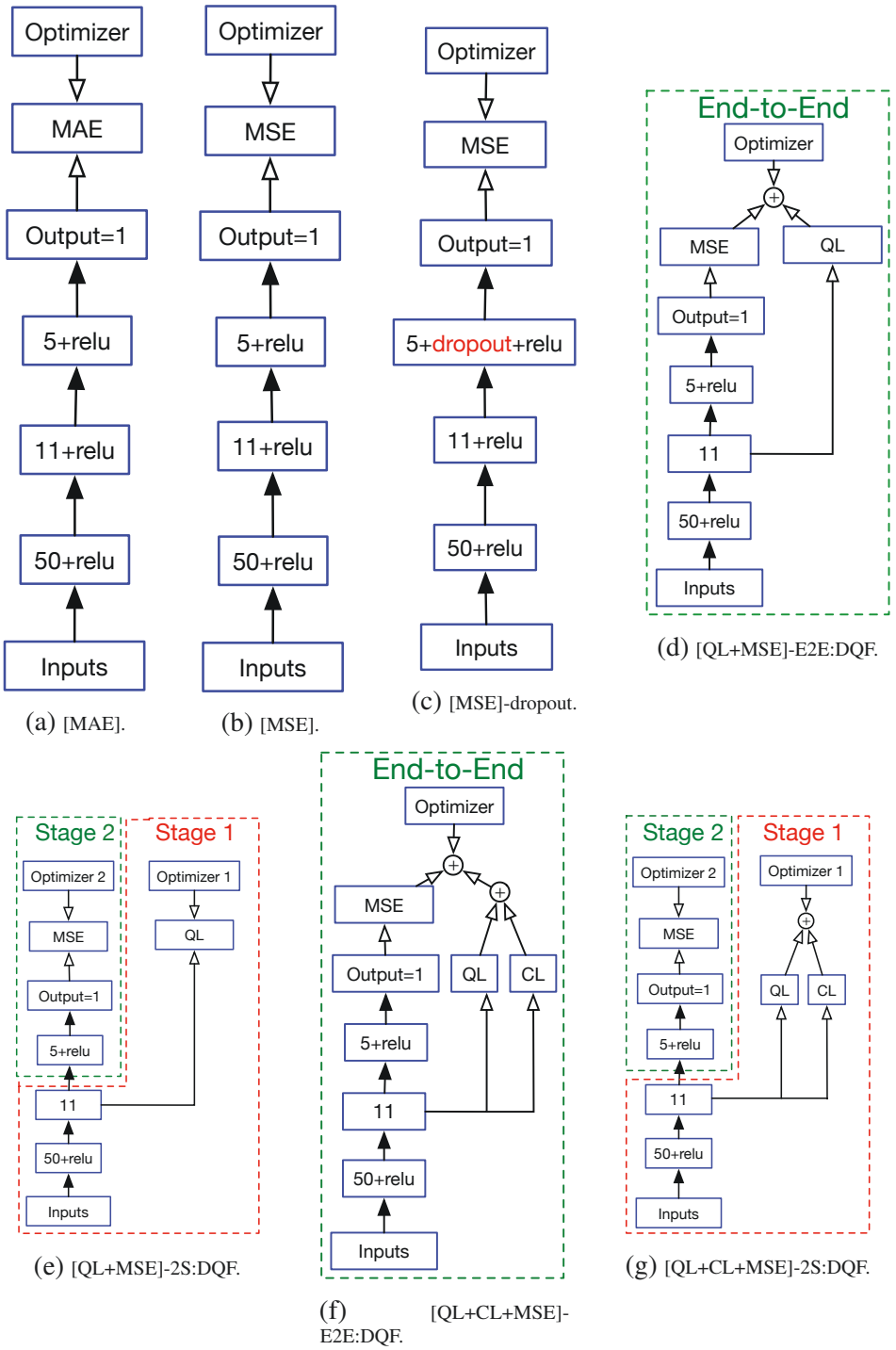


(c) [MSE].



(d) [QL+MSE]-E2E:LWQ.

Figure 6.2 DNN(50-11)-based and DeepJMQR models.



(a) [MAE]. (b) [MSE]. (c) [MSE]-dropout. (d) [QL+MSE]-E2E:DQF.

(e) [QL+MSE]-2S:DQF. (f) [QL+CL+MSE]-E2E:DQF. (g) [QL+CL+MSE]-2S:DQF.

Figure 6.3 DNN(50-11-5)-based models.

DeepJMQR This method jointly models the conditional mean and quantiles and adopts the mean output/estimation for forecasting (Rodrigues and Pereira, 2020). Its difference with DQF is that DeepJMQR does not take the quantiles as input for modeling the conditional mean. In our experiment, the setting of DeepJMQR has three hidden layers with 50 hidden nodes in each layer, which is shown in Figure 6.2a.

DNN(50-11)-based models:

- -[MAE]. It means training DNN(50-11) with MAE as the loss function, which is illustrated by Figure 6.2b.
- -[MSE]. It means training DNN(50-11) with MSE as the loss function, which is illustrated by Figure 6.2c.
- -[QL+MSE]-E2E:LWQ. It trains DNN(50-11) with QL plus MSE as the loss function in E2E mode. It directly connects 11 quantiles to the output layer for forecasting, which mathematically represents the forecasted target as a **L**inear **W**eighted based on the 11 **Q**uantiles (LWQ). Figure 6.2d shows its structure.

DNN(50-11-5)-based models:

- -[MAE]. It means training DNN(50-11-5) with MAE as the loss function, which is illustrated by Figure 6.3a.
- -[MSE]. It means training DNN(50-11-5) with MSE as the loss function, which is illustrated by Figure 6.3b.
- -[MSE]-Dropout. It has the same meaning as -[MSE] except adding a dropout layer, which is illustrated by Figure 6.3c. The Dropout rate is set to 0.2 in the

experiments. We also tried the dropout rates 0.5 and 0.8, but the result was much worse. The degeneration of dropout for deep regression is also reported in the past research (Lathuilière et al., 2020). Hence we would suggest not using the dropout strategy in forecasting tasks.

- -[QL+MSE]-E2E:DQF. It trains DNN(50-11-5) in E2E mode with MSE plus QL as the loss functions, which is illustrated by Figure 6.3d. Its advantage over DNN(50-11)-based [QL+MSE]-E2E:LWQ is that 6.2d [QL+MSE]:DQF further conducts a non-linear transformation by adding a 5-hidden-node layer to extract deeper distribution features from the learned 11 quantiles. It is hence so-called deep quantile fusion and improves performance dramatically. In practice, we can add more hidden nodes in the quantile layer to output more quantiles so as to describe the distribution even better, and meanwhile, extend the network N_2 deeper to extract robust features. Our experiments have shown the significant improvement of this design.

- -[QL+MSE]-2S:DQF. Similar to [QL+MSE]-E2E:DQF, but the model is optimized in 2S mode, which is illustrated by Figure 6.3e.

- -[QL+CL+MSE]-E2E:DQF. Similar to [QL+MSE]-E2E:DQF, but the model is optimized by three loss functions including QL, CL, and MSE together in E2E mode. Its architecture is illustrated in Figure 6.3f.

- -[QL+CL+MSE]-2S:DQF. Similar to [QL+CL+MSE]-E2E:DQF, but the model is optimized in 2S mode, which is illustrated by Figure 6.3e.

The setting of these methods can help us identify the performance of different methodologies. For example, by comparing [QL+MSE]-E2E:DQF (in Figure 6.3d) v.s. [MSE] (in Figure 6.3b), we can examine the effectiveness of DQF. By

comparing [QL+MSE]-E2E:DQF (in Figure 6.3d) v.s. [QL+MSE]-2S:DQF (in Figure 6.3e), we can observe the influence of E2E and 2S.

MC-Dropout We also re-run the open source codes of MC-Dropout which adopts Keras rather Pytorch as in the deep learning experiments (Gal and Ghahramani, 2016). Unfortunately, the experimental results are far less favourable than the results in the original paper (Gal and Ghahramani, 2016), which are summarized in Table 6.5. We can see that excepting that standard errors are similar, the means have a major distinction. Our explanatioin is that the difference in software and hardware environments can influence experimental results (Sandve et al., 2013). The reason we do not use Keras to code our DQF methodology is that Keras is a high-level library and cannot realize all methods in our experiments. Moreover, MC-Dropout deployed an exhausted grid-search strategy to find the best hyper parameters using validation datasets, which is not adopted in our DQF experiments. Our experiments simply train all models to 200 epochs and then apply them to do testing. All of this makes the comparison between MC-Dropout and our methods not equitable. It is important to emphasize that we aim to demonstrate the accuracy and robustness of the DQF methodology under the same training configurations, and we suppose that there is still substantial room for improvement in the performance of DQF by refining deep achitectures and hyper-parameter on specific datasets.

Table 6.1 RMSE evaluation. The ranking of five competitive methods, i.e., DNN (50-11-5)-based [QL+MSE]-E2E:DQF, [QL+CL+MSE]-E2E:DQF, [QL+MSE]-2S:DQF, [MSE] and DeepJMQR, are indicated at upper-right corner. The first-place one is also displayed in bold. The $\pm$X reported is standard error on 20 testing folds (not standard deviation).

Models	Boston Housing	Concrete	Energy	Yacht	Wine	Kin8m	Power	Protein
DNN (50-11-5)								
[MAE]	3.6862±0.4151	7.7707±0.8360	2.5635±0.0956	4.9575±1.0159	0.7935±0.0243	0.0765±0.0009	10.2380±1.4289	4.5750±0.0397
[MSE]	3.5422±0.3900$^{(10)}$	7.7052±0.8288$^{(10)}$	1.9145±0.1073$^{(2)}$	3.5312±0.5361$^{(4)}$	0.7537±0.0198$^{(8)}$	**0.0749±0.0009**$^{(1)}$	9.4885±1.4050$^{(10)}$	**4.2645±0.0054**$^{(1)}$
[MSE]-dropout	3.8769±0.3888	8.5511±0.8661	2.8576-±0.1860	4.8059±0.7470	0.7919±0.0151	0.0820±0.0017	10.7044±1.3151	4.3537±0.0246
[QL+MSE]-E2E:DQF	3.0894±0.1661$^{(2)}$	**5.8436±0.1041**$^{(1)}$	1.9231±0.1128$^{(3)}$	3.1299±0.3338$^{(3)}$	**0.6834±0.0115**$^{(1)}$	0.0768±0.0007$^{(4)}$	4.3935±0.0306$^{(2)}$	4.3899±0.0160$^{(4)}$
[QL+MSE]-2S:DQF	**3.0724±0.1779**$^{(1)}$	5.9881±0.1070$^{(2)}$	2.2627±0.0766$^{(7)}$	**2.9911±0.2150**$^{(1)}$	0.6896±0.0101$^{(3)}$	0.0816±0.0007$^{(8)}$	4.4218±0.0321$^{(4)}$	4.5297±0.0169$^{(6)}$
[QL+CL+MSE]-E2E:DQF	3.2778±0.2230$^{(7)}$	6.1563±0.1490$^{(5)}$	2.1062±0.1023$^{(5)}$	3.5395±0.3359$^{(5)}$	0.6893±0.0124$^{(2)}$	0.0781±0.0007$^{(6)}$	**4.3845±0.0303**$^{(1)}$	4.4234±0.0346$^{(2)}$
[QL+CL+MSE]-2S:DQF	3.2806±0.1911	6.2021±0.1371	2.4574±0.0692	4.5514±0.8285	0.6897±0.0129	0.0805±0.0009	4.3961±0.0303	4.5687±0.0202
DNN (50-11)								
[MAE]	3.2381±0.2484	6.2169±0.1255	2.5494±0.0889	5.5494±0.9747	0.7098±0.0264	0.0772±0.0009	7.6959±1.2519	4.6175±0.0332
[MSE]	3.1319±0.1936	6.1253±0.1282	1.9828±0.0983	3.7017±0.5322	0.7216±0.0170	0.0754±0.0010	6.9789±1.1200	4.3637±0.0259
[QL+MSE]-E2E:LWQ	3.0958±0.1918	6.0548±0.1055	2.1684±0.0839	4.8754±0.6052	0.7272±0.0186	0.0810±0.0007	4.4891±0.0650	4.5210±0.0175
DeepJMQR	3.2100±0.1243$^{(5)}$	6.8701±0.1309$^{(8)}$	**0.8419±0.0406**$^{(1)}$	3.0458±0.1935$^{(2)}$	0.8800±0.0115$^{(11)}$	0.0883±0.0009$^{(11)}$	5.9189±0.0495$^{(6)}$	5.3016±0.0287$^{(11)}$

Table 6.2 MAE evaluation. Note that the ranking of methods are not consistent as RMSE evaluation since both two evaluation are not linearly correlated.

Models	Boston Housing	Concrete	Energy	Yacht	Wine	Kin8m	Power	Protein
DNN (50-11-5)								
[MAE]	2.5220±0.2895	5.8805±0.7119	1.5578±0.0637	3.0274±0.5809	0.5828±0.0228	0.0594±0.0006	8.5555±1.2467	**3.0327±0.0223**
[MSE]	2.5119±0.2850$^{(9)}$	5.9768±0.7026$^{(11)}$	1.4562±0.0721$^{(3)}$	2.3297±0.4023$^{(5)}$	0.6202±0.0211$^{(9)}$	**0.0584±0.0008**$^{(1)}$	8.0387±1.2598$^{(10)}$	3.1839±0.0149$^{(3)}$
[MSE]-dropout	2.7643±0.2804	6.7019±0.7251	2.3922±0.1374	3.3086±0.5030	0.6689±0.0145	0.0642±0.0011	9.1058±1.1843	3.3722±0.0308
[QL+MSE]-E2E:DQF	2.1360±0.0712$^{(2)}$	**4.3941±0.0811**$^{(1)}$	1.4163±0.0837$^{(2)}$	1.9269±0.1840$^{(2)}$	0.5333±0.0083$^{(2)}$	0.0600±0.0005$^{(5)}$	**3.4628±0.0150**$^{(1)}$	3.3293±0.0248$^{(6)}$
[QL+MSE]-2S:DQF	**2.1221±0.0745**$^{(1)}$	4.4951±0.0864$^{(2)}$	1.5055±0.0566$^{(6)}$	2.0734±0.1416$^{(3)}$	0.5424±0.0108$^{(6)}$	0.0627±0.0005$^{(9)}$	3.4921±0.0213$^{(4)}$	3.4990±0.0175$^{(8)}$
[QL+CL+MSE]-E2E:DQF	2.2933±0.0910$^{(7)}$	4.6156±0.1105$^{(6)}$	1.4734±0.0633$^{(5)}$	2.2318±0.1951$^{(4)}$	0.5346±0.0113$^{(3)}$	0.0608±0.0005$^{(6)}$	3.4629±0.0177$^{(2)}$	3.3117±0.0338$^{(5)}$
[QL+CL+MSE]-2S:DQF	2.2793±0.0982	4.6436±0.1029	1.6114±0.0577	3.1790±0.5958	0.5373±0.0097	0.0626±0.0007	3.4644±0.0178	3.4698±0.0280
DNN (50-11)								
[MAE]	2.2427±0.1053	4.5509±0.0896	1.5112±0.0703	3.2149±0.5135	**0.5245±0.0199**	0.0597±0.0007	6.3008±1.0960	3.1515±0.0336
[MSE]	2.2100±0.0948	4.6133±0.1041	1.4613±0.0682	2.5931±0.4276	0.5757±0.0174	0.0590±0.0007	5.7637±1.0007	3.3037±0.0249
[QL+MSE]-E2E:LWQ	2.2162±0.1065	4.5795±0.0906	1.5617±0.0637	3.7955±0.4711	0.5782±0.0159	0.0630±0.0006	3.5404±0.0504	3.5352±0.0386
DeepJMQR	2.3501±0.0606$^{(8)}$	5.0713±0.0992$^{(8)}$	**0.6259±0.0294**$^{(1)}$	**1.7910±0.1124**$^{(1)}$	0.6599±0.0095$^{(10)}$	0.0691±0.0007$^{(11)}$	4.6895±0.0396$^{(6)}$	3.6900±0.0223$^{(11)}$

Table 6.3 Top-n hits of five competitive methods on eight datasets.

	RMSE			MAE		
DNN (50-11-5)	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
[QL+MSE]-E2E:DQF	2*	6*	8*	2*	6*	7*
[QL+CL+MSE]-E2E:DQF	1	3	6	0	2	5
[QL+MSE]-2S:DQF	2*	4	6	1	3	4
[MSE]	2*	3	4	1	3	4
DeepJMQR	1	2	3	2*	2	2

Table 6.4 An example statistic of quantile crossing on one testing data fold.

Dataset	Testing data size	Crossing numbers (o/w CL)^2	Crossing ratio (o/w CL)
bostonHousing	51	49/12	0.96/0.24
Concrete	103	99/29	0.96/0.28
Energy	77	73/27	0.95/0.35
Yacht	31	31/30	1.00/0.97
Wine	160	151/4	0.94/0.03
Kin8m	819	488/71	0.60/0.09
Power	957	83/2	0.09/0.00
Protein	4573	3631/1187	0.79/0.26

Table 6.5 The evaluation metric is RMSE, N is the number of testing data and D is the number of input feature. From left to right, it is MC-Dropout (Keras+Theano) (Gal and Ghahramani, 2016), MC-Dropout (Keras+Tensorflow) (our reproduction), DeepEnsemble (Lakshminarayanan et al., 2017), and PBP (Hernández-Lobato and Adams, 2015).

Dataset	N	D	MC-Dropout (Keras+Theano)	*MC-Dropout (Keras+Tensorflow)*	DeepEnsemble (Torch)	PBP (Theano)
Boston Housing	506	13	2.97±0.19	*4.02±0.23*	3.28±1.00	3.01±0.18
Concrete	1030	8	5.23±0.12	*8.24±0.13*	6.03±0.58	5.67±0.09
Energy	768	8	1.66±0.04	*2.95±0.06*	2.09±0.29	1.80±0.05
Yacht	308	6	1.11±0.09	*9.08±0.39*	1.58±0.48	1.02±0.05
Wine	1599	11	0.62±0.01	*0.64±0.01*	0.64±0.04	0.64±0.01
Kin8m	8192	8	0.10±0.00	*0.11±0.00*	0.09±0.00	0.10±0.00
Power	9586	4	4.02±0.04	*4.19±0.03*	4.11±0.17	4.12±0.03
Protein	45730	9	4.36±0.01	*4.64±0.01*	4.71±0.06	4.73±0.01

6.3.3 Experimental results and analysis

We evaluate the forecasting accuracy by RMSE and MAE metrics:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{6.9}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{6.10}$$

where N is the testing data size for each UCI dataset which is shown in Table 6.3. The evaluation results are summarized in Table 6.1 and Table 6.2. We mark the ranking in the upper-right corner particularly for five competitive methods including DNN (50-11-5)-based [QL+MSE]-E2E:DQF, [QL+CL+MSE]-E2E:DQF, [QL+MSE]-2S:DQF, [MSE] and DeepJMQR. The first place one is also displayed in bold. Please note the $\pm$X reported is standard error (not standard deviation). In general, models trained by MAE or MSE alone have poor performance, while models trained with the DQF methodology have better generalization and robustness. We elaborate the analysis as following.

- **Effectiveness of DQF.** Generally, models with DQF methodology (with *:DQF* suffix) perform better. It is noticeable that DNN (50-11-5)-based [QL+MSE]-E2E:DQF performs much better than models [MAE] and [MSE]. On some datasets, the method [MSE] is competitive, but its performance is not always robust on other datasets e.g., Boston Housing and Concrete. In the beginning, we thought that the method [MSE] had been over-fitted so therefore we made the baseline [MSE]-dropout to validate our thinking. However, [MSE]-dropout performed even worse, and we concluded that the poor performance of the model [MSE] was not due to over-fitting.

[QL+MSE]-E2E:DQF also nearly beat DeepJMQR except for in the dataset Energy. There is no one model which can always perform best on every dataset. We hence summarize the hits of top-1, top-3, and top-5 on eight datasets for the five most competitive methods shown in Table 6.3. Overall, we can see that [QL+MSE]-E2E:DQF always achieves the most hits (* marked) for every top-n, and achieves considerable performance no matter if evaluated by RMSE or MAE.

- **Robustness of DQF.** DNN (50-11-5)-based [QL+MSE]-E2E:DQF is noticeably robust. For example, on dataset Concrete, it achieved rank-1 RMSE 5.8436 ± 0.1041 but [MSE] only achieved rank-10 RMSE 7.7052 ± 0.8288, and the similar performance can be seen on the Boston Housing dataset. Conversely, although [MSE] had the best RMSE performance 0.0749 ± 0.0009 on Kin8m, [QL+MSE]-E2E:DQF still achieved a competitive RMSE 0.0768 ± 0.0007, and the same performance can be seen on dataset Protein. It should be noted that the forecasting performance on datasets Kin8m and Protein have small standard error and most methods have close competitive performance. Conversely, forecasting performance on datasets such Concrete, Boston Housing and Power have large standard error where methods without DQF perform poorly. Our interpretation is that no matter what the variance of a dataset, DQF methodology can adaptively learn distribution information from the quantiles and hence behaves robustly.

- **Linear Weighted Quantile v.s. Deep Quantile Fusion.** By comparing DNN (50-11)-based [QL+MSE]-E2E:LWQ and DNN (50-11-5)-based [QL+MSE]-E2E:DQF, we can see that DQF brings a huge improvement for accuracy and robustness. On the other hand, among DNN(50-11)-based models,

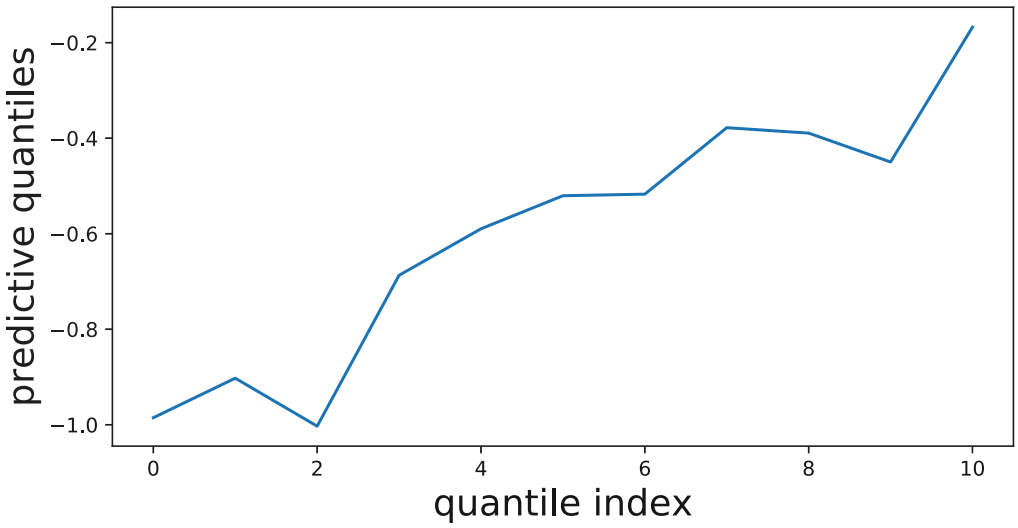
[QL+MSE]-E2E:LWF has no significant improvement than [MAE] and [MSE], which further indicates the importance of deep quantile fusion.

- **End-to-End v.s. Two Stages.** By comparing [QL+MSE]-E2E:DQF v.s. [QL+MSE]-2S:DQF and [QL+CL+MSE]-E2E:DQF v.s. [QL+CL+MSE]-2S:DQF, our conclusion is that training in E2E mode is better than in 2S mode for the forecasting target.

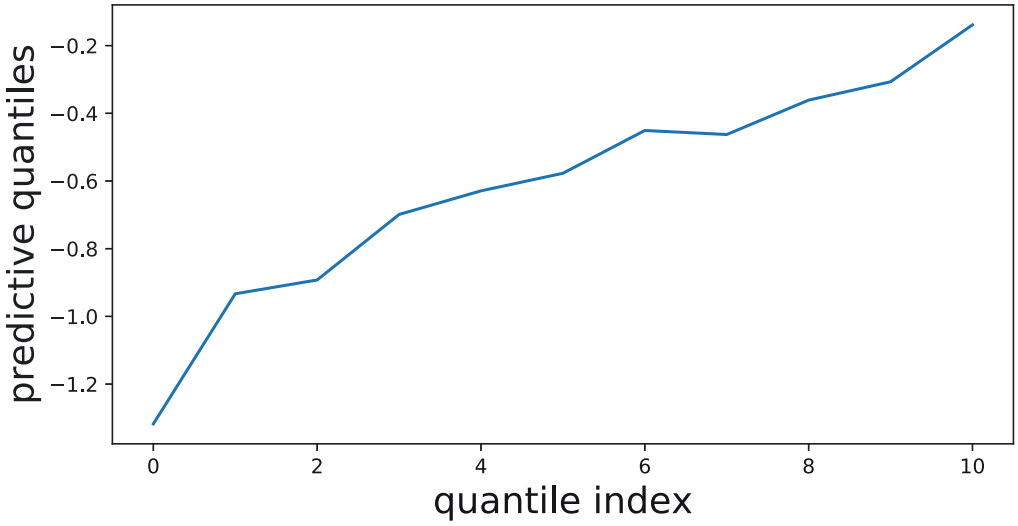
- **Effectiveness of Crossing Loss.** Table 6.4 summarized the results of quantile crossing when we use [QL+MSE]-E2E and [QL+CL+MSE]-E2E to test. Even when training deep models with CL, it is logical that quantile crossing can sometimes happen. It should be emphasized that this study concerns how to exploit learned quantiles to improve forecasting accuracy rather than the QR itself.

Figure 6.4 shows the results of two forecasting models for the same testing example. In Figure 6.4a, the line is not monotonic, which means quantile crossing does not happen. In Figure 6.4a, the line is monotonic, which means quantile crossing happens. Generally, models trained with CL can significantly decrease the quantile crossing. In Figure 6.5, we can see that output quantiles are very separate, which is predicted by the model [QL+CL+MSE]-E2E.

Overall, we conclude that optimizing deep models by QL plus MSE via E2E learning, i.e., method [QL+MSE]-E2E:DQF, best matches the DQF considering the forecasted target.



(a) The output quantiles of the model [QL+MSE]-E2E:DQF.

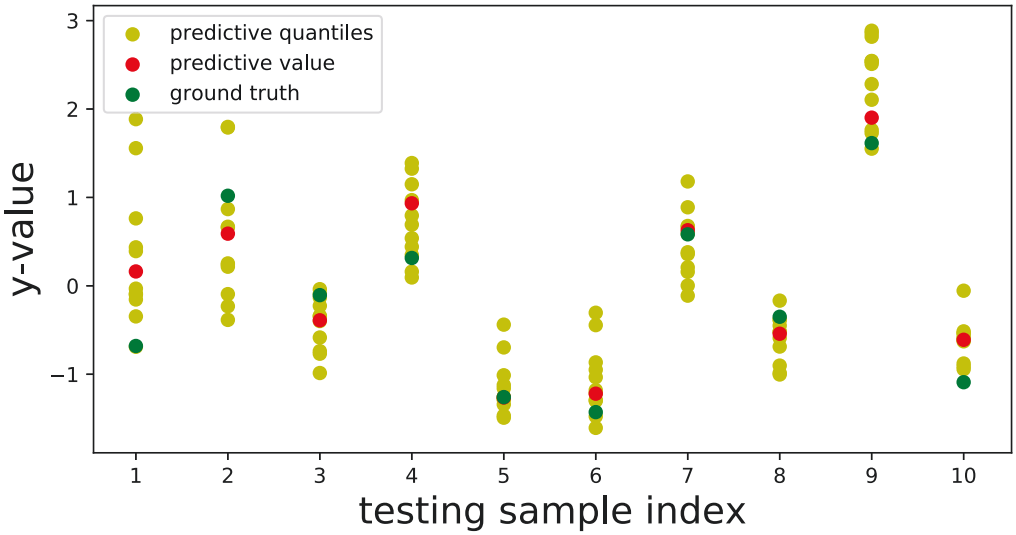


(b) The output quantiles of the model [QL+CL+MSE]-E2E:DQF.

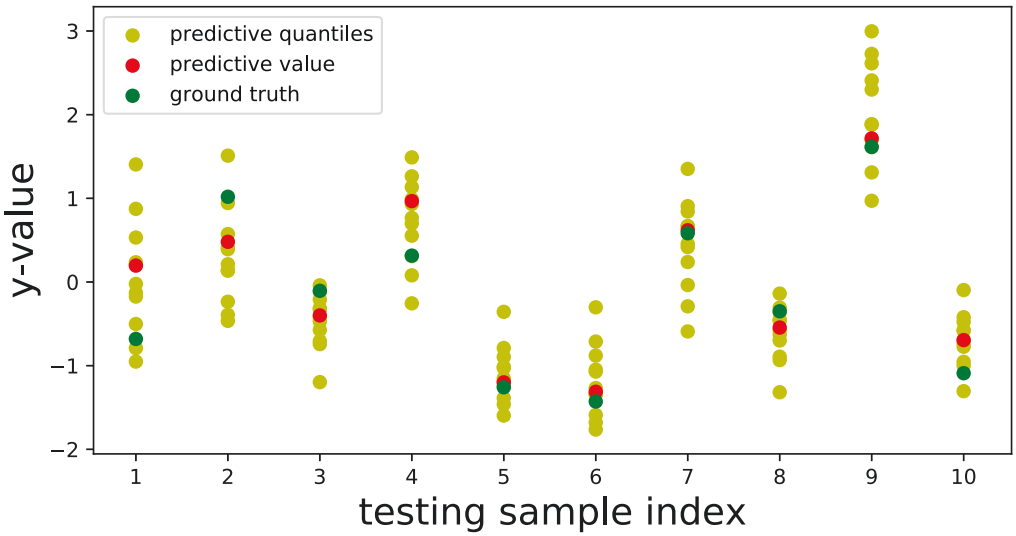
Figure 6.4 The output quantiles of two models trained o/w CL.

6.4 Model Extension

In this section, we illustrate how to apply quantile fusion into AutoEncoder, built as QF-AE, to model time series.



(a) Outputs of [QL+MSE]-E2E.



(b) Outputs of [QL+CL+MSE]-E2E.

Figure 6.5 10 output samples on Concrete dataset. For each sample, the yellow points represent 11 quantiles, the green points are the ground truth and the red points are the regressed/predictive values.

6.4.1 Deep auto-encoder with quantile fusion

We construct a feedforward neural network to introduce the model framework, which is shown in Figure 6.6. Sequential models such as LSTM can be designed

accordingly. The nodes of the input layer are set as the time steps of the series, 10 in our case. All hidden layers are set with 40 nodes and *tanh* activations. In the quantile layer, it consists of 10 pairs of [5% quantile, 95% quantile], hence 20 nodes, which predict the 5% quantile and 95% quantile for each of 10 time steps. Any other quantiles for any number of time steps can also be accomplished accordingly. The output of the quantile layer is optimized using quantile loss and is concatenated as the input for the following layer. The output layer will reconstruct the input series by MSE loss function, and the entire model is trained via QL+MSE referred to the formula (6.7).

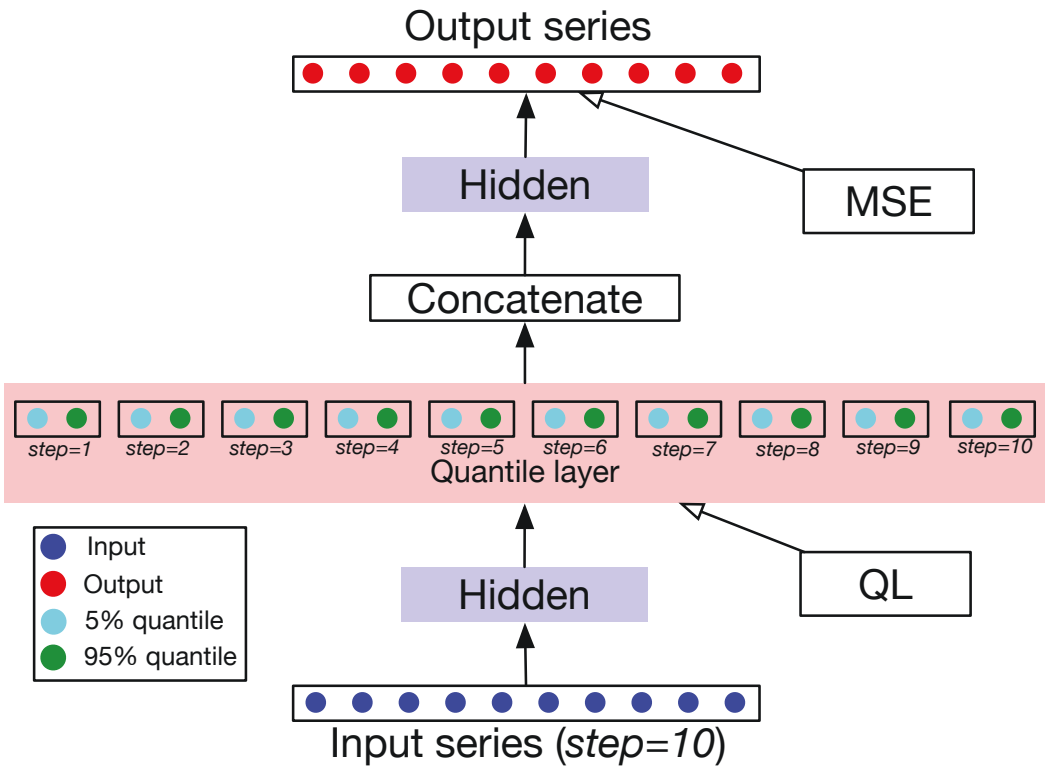


Figure 6.6 The framework of quantile fusion-based AutoEncoder.

6.4.2 Dataset

The experimental UCI dataset includes a total of 10,000 synthetic time series [3]. Each instance is a 10-step time series ranging from -0.5 to 0.5. We train the model on 50000 instances, and take a left 50000 for the test.

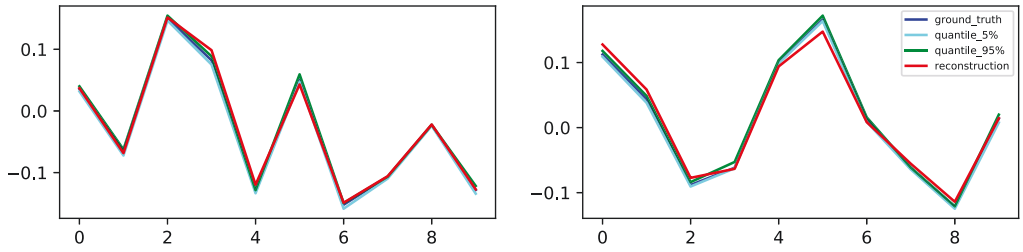
6.4.3 Experimental analysis

The training batch size is set as 1024. After training 20 epochs on the training set, we evaluate the reconstruction error on the test set by MSE and MAE criterion. Note that the baseline AE model is built with the same structure as QF-AE, yet it is only trained by MSE loss function. Table 6.6 presents that QF-AE achieved the smaller reconstruction error (MSE: 9.83e-5, MAE: 7.45e-3) compared to the baseline AE (MSE: 1.53e-4, MAE: 9.61e-3). This indicates quantile fusion can help AutoEncoder extract effective representations to decrease reconstruction error. Figure 6.7 displays reconstruction examples with the predefined quantiles. We define a measurement to retrieve the unsatisfactory reconstruction if the reconstructed sequence falls outside the 90% prediction interval up to six times.

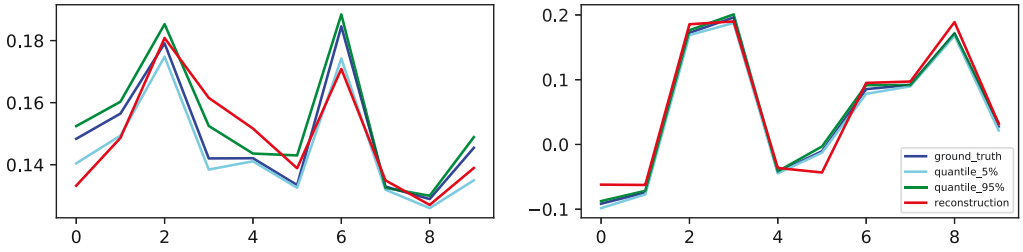
Table 6.6 Reconstruction error evaluation.

	Test MSE	Test MAE
Baseline AE	1.53e-4	9.61e-3
QF-AE	9.83e-5	7.45e-3

[3] https://archive.ics.uci.edu/ml/datasets/Pseudo+Periodic+Synthetic+Time+Series



(a) Examples of good reconstruction. The reconstructed sequence nearly falls inside the 90% prediction interval.



(b) Examples of unsatisfactory reconstruction. The reconstructed sequence frequently falls outside the 90% prediction interval.

Figure 6.7 Examples of reconstructing 10-step time series.

6.5 Summary

In this chapter, we proposed a generic methodology to achieve more accurate and robust performance for forecasting problems. It first conducted QR to learn quantiles and then took these quantiles as input to forecast the targeted value. By this, it can extract robust distribution features from the learned quantiles. We then introduced a crossing loss function to alleviate quantile crossing and compare the performance of two learning modes. We have evaluated our method on eight UCI datasets and shown its effectiveness. Overall, [QL+MSE]-E2E:DQF which trains deep models by QL plus MSE in E2E mode has achieved the best performance. We further proposed to incorporate quantile fusion in AutoEncoder and demonstrated its superiority when modeling time series.

In the future, we will polish [QL+CL+MSE]-E2E:DQF to achieve better performance. For example, we can try to improve it to achieve the same performance as the method [QL+MSE]-E2E:DQF while reducing quantile crossing. We will explore its performance in spatio-temporal forecasting scenarios by extending QF using a sequential model, developing an attention mechanism (Razzak et al., 2019), and introducing interpretable techniques (Assaf et al., 2019).

Chapter 7

Conclusion and Future Research

This chapter concludes the thesis and provides further research directions for this topic.

7.1 Conclusions

Time series forecasting has always been a hot issue in scientific research. The automatic feature extract and powerful representative ability of deep learning make it an important learning machine for handling time series forecasting tasks. This thesis has proposed novel point estimation and uncertainty quantification methods based on deep learning techniques. It first focused on the spatio-temporal point estimation problem and then introduced two deep uncertainty quantification methods. Finally, it designed a deep quantile fusion method, which considered taking advantage of uncertainty quantization to help improve the accuracy of point estimation. The main contributions of this thesis are summarized as follows:

1. Aiming at the problem that deep learning has a long training time in point estimation and weak generalization ability of a single model, a convolutional

neural network Noise-ResNet with noisy residual connections and a snapshot ensemble prediction method FUSE are designed. The Noise-ResNet method improves the model's generalization ability for point estimation in high-dimensional predictions by introducing noise perturbations into the residual connections as a regularization. This study takes the typical application of spatiotemporal flow forecasting in the area as an example. By selecting the input data reasonably, the complexity of the model is appropriately reduced, so that the model can greatly reduce the training time while ensuring the generalization accuracy. This study also verifies the superior performance of the snapshot ensemble method FUSE in the prediction task, which can greatly improve the prediction accuracy while the training time of the single model is not much different. Experiments are evaluated on two real traffic flow datasets, and the results demonstrate the effectiveness of the proposed method.

2. Aiming at how to quantify the stochastic uncertainty in prediction, a deep uncertainty quantification method DUQ with distribution assumptions is proposed. This method adopts the likelihood loss function based on the assumption of Gaussian distribution as the training target of the Encoder-Decoder model, thereby realizing the prediction function of point estimation and interval estimation for multiple times and variables in the future. This study is the first to report the generalization improvement of deep prediction models with likelihood loss function as training target. In this study, the method is applied to the actual weather forecast problem. By designing an effective multi-source information fusion mechanism, the forecast accuracy of various meteorological elements can be improved, and the forecast interval

can also be estimated. This research has certain reference value and prospect for the application of deep learning to deal with meteorological forecasting problems.

3. To further solve the problem that the true distribution of the target variables in practice cannot be foretold, a deep uncertainty quantification method without distribution assumption is proposed. The method constructs a novel loss function without distribution hypothesis, the so-called distribution-free loss function, as the objective function of training the encoder-decoder to capture data distribution adaptively in the training stage and then output point estimation and prediction interval in the forecasting stage. The effectiveness of the proposed method is verified on three public datasets.

4. Aiming at the problems of low accuracy and weak robustness in prediction, a novel deep quantile fusion method for point estimation is developed. By designing a novel loss function, the method can transform the hidden layer into a quantile layer with semantic information and take quantiles as the input feature of the following layer to forecast the target variable, thereby constructing a deep forecasting model with high accuracy and great robustness. The experiments are evaluated on eight UCI regression datasets and one time series dataset, and the results validate the effectiveness of the proposed methods.

7.2 Future Study

Although this study involves several contributions to the advancement of deep forecasting, some experts argue that deep learning is almost a black box model

hence lacking interpretability and theoretical support. Therefore, there are still many improvements and concerns that need to be addressed in the deep forecasting research. The following research directions could serve as worthwhile future study:

- Interpretability of deep forecasting. With the general applications of deep learning, the interpretability of deep forecasting models has become increasingly prominent. Interpretable deep learning will significantly improve the reliability of the predictive results, making them knowable, credible, and controllable. There are relatively few interpretability studies on deep forecasting, and they mainly focus on the interpretation of point estimation (Li et al., 2019; Siddiqui et al., 2020; Siddiqui et al., 2019). The following research directions would be to conduct interpretation research of both point estimation and uncertainty quantification, which can guide operators to figure out what input features influence the point estimation and what input features involve the uncertainty degree. Such interpretability not only advises model operators on the information for decision making but also has excellent potential for even better data augmentation.

- Ensemble of deep forecasting. Ensemble learning is a critical technique to improve model generalization and overcome the phenomenon of high variance. Currently, the research on deep ensemble is mainly oriented to the computer vision problems (Wasay et al., 2018). There are few studies on how to devise the deep forecasting ensemble techniques by considering the characteristics of the time series data (Lakshminarayanan et al., 2017). In Chapter 2 and Chapter 3, this thesis introduces a naive but time-consuming ensemble method by averaging multiple deep models and a remarkably time-saving method FUSE. Both experimental results indicate that deep ensemble

can dramatically improve the generalization of deep forecasting. Building more advanced ensembled deep forecasting methods and theories will be an auspicious research direction. Moreover, the object to be ensembled need not be completely limited to point estimation but can also be extended to uncertainty quantification so that the quality of deep uncertainty quantification can be further enhanced via ensemble.

- Standardization of datasets. Although there is massive public time series and spatio-temporal data, they are non-preprocessed (raw) but not preprocessed-well standard datasets that can be directly taken as input to machine learning models. A specific time series forecasting task, often requires unique data exploration, missing value imputation, exceptional value processing, feature extraction, and other preprocessing procedures. For the sake of data privacy and protection, many studies only expose the source code, or at most a small part of the sample data; sometimes, different studies certainly use the same original dataset, but distinct data preprocessing procedures are adopted. Such a procedure undoubtedly can decrease the credibility of experiment reproducibility and model comparability (Hutson, 2018). In the future study, it would be worth providing benchmark datasets with uniform preprocessed steps and utilizing tuple formation to describe the shape of input and output data. This formatted expression of data shape is convenient for researchers to understand the data size and feature dimension quickly and how to transform benchmark datasets for the particular machine learning model, hence allowing researchers to focus more on model design and comparisons.

Bibliography

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297.

Aigner, W., Miksch, S., Schumann, H., and Tominski, C. (2011). *Visualization of time-oriented data*.

Assaf, R., Giurgiu, I., Bagehorn, F., and Schumann, A. (2019). Mtex-CNN: multivariate time series explanations for predictions with convolutional neural networks. In *International Conference on Data Mining (ICDM)*, pages 952–957. IEEE.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. (2020). Neural forecasting: Introduction and literature overview.

Benvenuto, D., Giovanetti, M., Vassallo, L., Angeletti, S., and Ciccozzi, M. (2020). Application of the arima model on the covid-2019 epidemic dataset. *Data in brief*, 29:105340.

Bundy, A. (2017). Preparing for the future of artificial intelligence. *AI & SOCIETY*, 32(2):285–287.

Candanedo, L. M., Feldheim, V., and Deramaix, D. (2017). Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97.

Chen, C., Li, K., Teo, S. G., Chen, G., Zou, X., Yang, X., Vijay, R. C., Feng, J., and Zeng, Z. (2018a). Exploiting spatio-temporal correlations with multiple 3D convolutional neural networks for citywide vehicle flow prediction. In *International Conference on Data Mining (ICDM)*, pages 893–898.

Chen, L. and Lai, X. (2011). Comparison between ARIMA and ANN models used in short-term wind speed forecasting. In *Power and Energy Engineering Conference (APPEEC)*, pages 1–4. IEEE.

Chen, T., Yin, H., Chen, H., Wu, L., Wang, H., Zhou, X., and Li, X. (2018b). Tada: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction. In *International Conference on Data Mining (ICDM)*, pages 49–58.

Chen, X., Chen, H., Yang, Y., Wu, H., Zhang, W., Zhao, J., and Xiong, Y. (2021). Traffic flow prediction by an ensemble framework with data denoising and deep learning model. *Physica A: Statistical Mechanics and Its Applications*, 565:125574.

Chen, Y., Xu, P., Chu, Y., Li, W., Wu, Y., Ni, L., Bao, Y., and Wang, K. (2017). Short-term electrical load forecasting using the support vector regression (svr) model to calculate the demand response baseline for office buildings. *Applied Energy*, 195:659–670.

Choudhary, S., Hiranandani, G., and Saini, S. K. (2018). Sparse decomposition for time series forecasting and anomaly detection. In *SIAM International Conference on Data Mining (SIAM)*, pages 522–530.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2018). Distributional reinforcement learning with quantile regression. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 2892–2901.

De Gooijer, J. G. and Hyndman, R. (2005). 25 years of IIF time series forecasting: A selective review. *SSRN Electronic Journal*.

Deng, D., Shahabi, C., Demiryurek, U., and Zhu, L. (2017). Situation aware multi-task learning for traffic prediction. In *International Conference on Data Mining (ICDM)*, pages 81–90.

Deng, Z., Wang, B., Guo, H., Chai, C., Wang, Y., Zhu, Z., and Cai, N. (2020). Unified quantile regression deep neural network with time-cognition for probabilistic residential load forecasting. *Complex.*, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding.

Diao, Z., Wang, X., Zhang, D., Liu, Y., Xie, K., and He, S. (2019). Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 890–897.

Doctorow and Cory (2008). Big data: welcome to the petacentre. *Nature*, 455(7209):16–21.

Faloutsos, C., Gasthaus, J., Januschowski, T., and Wang, Y. (2018). Forecasting big time series: old and new. *Proceedings of the VLDB Endowment*, 11(12):2102–2105.

Fang, T. and Lahdelma, R. (2016). Evaluation of a multiple linear regression model and sarima model in forecasting heat demand for district heating system. *Applied energy*, 179:544–552.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery*, 33:917–963.

Feng, Y., Xu, Z., Gan, L., Chen, N., Yu, B., Chen, T., and Wang, F. (2019). DCmn: double core memory network for patient outcome prediction with multimodal data. In *International Conference on Data Mining (ICDM)*, pages 200–209. IEEE.

Fortunato, M., Blundell, C., and Vinyals, O. (2017). Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pages 1050–1059.

Gao, J., Wang, X., Wang, Y., Yang, Z., Gao, J., Wang, J., Tang, W., and Xie, X. (2019). Camp: Co-attention memory networks for diagnosis prediction in healthcare. pages Quantile regression neural networks1036–1041.

Geng, X., Li, Y., Wang, L., Zhang, L., Yang, Q., Ye, J., and Liu, Y. (2019). Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 3656–3663.

Gers, F. A., Eck, D., and Schmidhuber, J. (2002). Applying LSTM to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*, pages 193–200. Springer.

Ghaderi, A., Sanandaji, B. M., and Ghaderi, F. (2017). Deep forecast: deep learning-based spatio-temporal forecasting. *arXiv preprint arXiv:1707.08110*.

Gneiting, T. and Raftery, A. E. (2005). Weather forecasting with ensemble methods. *Science*, 310(5746):248–249.

Grover, A., Kapoor, A., and Horvitz, E. (2015). A deep hybrid model for weather forecasting. In *International Conference on Knowledge Discovery and Data Mining (ICDM)*, pages 379–386. ACM.

Gulcehre, C., Moczulski, M., Denil, M., and Bengio, Y. (2016). Noisy activation functions. In *International Conference on Machine Learning (ICML)*, pages 3059–3068.

Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. (2019). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 922–929.

Guo, T., Bifet, A., and Antulov-Fantulin, N. (2018). Bitcoin volatility forecasting with a glimpse into buy and sell orders. In *International Conference on Data Mining (ICDM)*, pages 989–994.

Gupta, M., Gao, J., Aggarwal, C., and Han, J. (2014). Outlier detection for temporal data: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267.

Han, M., Zhang, S., Xu, M., Qiu, T., and Wang, N. (2018). Multivariate chaotic time series online prediction based on improved kernel recursive least squares algorithm. *IEEE Transactions on Cybernetics*, 49(4):1160–1172.

Haslbeck, J. M., Bringmann, L. F., and Waldorp, L. J. (2021). A tutorial on estimating time-varying vector autoregressive models. *Multivariate Behavioral Research*, 56(1):120–149.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645.

He, Y., Li, H., Wang, S., and Yao, X. (2021). Uncertainty analysis of wind power probability density forecasting based on cubic spline interpolation and support vector quantile regression. *Neurocomputing*, 430:121–137.

Helbing, D., Brockmann, D., Chadefaux, T., Donnay, K., Blanke, U., Woolley-Meza, O., Moussaid, M., Johansson, A., Krause, J., Schutte, S., et al. (2015). Saving human lives: what complexity science and information systems can contribute. *Journal of Statistical Physics*, 158(3):735–781.

Hernández, E., Sanchez-Anguix, V., Julian, V., Palanca, J., and Duque, N. (2016). Rainfall prediction: a deep learning approach. In *International Conference on Hybrid Artificial Intelligence Systems (HAIS)*, pages 151–162. Springer.

Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning (ICML)*, pages 1861–1869.

Hewamalage, H., Bergmeir, C., and Bandara, K. (2019). Recurrent neural networks for time series forecasting: Current status and future directions.

Hoang, M. X., Zheng, Y., and Singh, A. K. (2016). Fccf: forecasting citywide crowd flows based on big data. In *International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, page 6.

Hong, T., Xie, J., and Black, J. (2019). Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting. *International Journal of Forecasting*, 35(4):1389–1399.

Hu, A., Jiao, Y., Liu, Y., Shi, Y., and Wu, Y. (2021). Distributed quantile regression for massive heterogeneous data. *Neurocomputing*, 448:249–262.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017). Snapshot ensembles: Train 1, get M for free. In *International Conference on Learning Representations (ICLR)*.

Hutson, M. (2018). Artificial intelligence faces reproducibility crisis. *Science*, 359(6377):725–726.

Hyndman, R. J. and Athanasopoulos, G. (2014). *Forecasting: Principles and practice*.

Ian Goodfellow, Y. B. and Courville, A. (2016). Deep learning.

Johnson, M. and Willsky, A. (2014). Stochastic variational inference for Bayesian time series models. In *International Conference on Machine Learning (ICML)*, pages 1854–1862.

Ke, J., Zheng, H., Yang, H., and Chen, X. M. (2017). Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation Research Part C: Emerging Technologies*, 85:591–608.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2010). Lower upper bound estimation method for construction of neural network-based prediction intervals. *Transactions on Neural Networks*, 22(3):337–346.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2011a). Comprehensive review of neural network-based prediction intervals and new advances. *Transactions on Neural Networks*, 22(9):1341–1356.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2011b). Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on Neural Networks*, 22(9):1341–1356.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2011c). Lower upper bound estimation method for construction of neural network-based prediction intervals. *Transactions on Neural Networks*, 22(3):337–346.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kodba, S., Perc, M., and Marhl, M. (2005). Detecting chaos from a time series. *European Journal of Physics*, 26(1):205–215.

Koenker, R. and Bassett Jr, G. (1978). Regression quantiles. *Econometrica: Journal of the Econometric Society*, pages 33–50.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6402–6413.

Lathuilière, S., Mesejo, P., Alameda-Pineda, X., and Horaud, R. (2020). A comprehensive analysis of deep regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2065–2081.

Li, H., Shen, Y., and Zhu, Y. (2018). Stock price prediction using attention-based multi-input LSTM. In *Asian Conference on Machine Learning (ACML)*, pages 454–469.

Li, L., Yan, J., Yang, X., and Jin, Y. (2019). Learning interpretable deep state space model for probabilistic time series forecasting. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2901–2908.

Li, Y. and Yuan, Y. (2017). Convergence analysis of two-layer neural networks with relu activation. *Advances in neural information processing systems*, 30.

Liao, Q. and Poggio, T. (2016). Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*.

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764.

Lim, B. and Zohren, S. (2020). Time series forecasting with deep learning: a survey.

Lin, Z., Feng, J., Lu, Z., Li, Y., and Jin, D. (2019). DeepSTN+: context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 1020–1037.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. (2021). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*.

Liu, Z. (2010). Chaotic time series analysis. *Mathematical Problems in Engineering*, 2010:1–31.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., and Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363.

Lu, J., Xuan, J., Zhang, G., and Luo, X. (2018). Structural property-aware multilayer network embedding for latent factor analysis. *Pattern Recognition*, 76:228–241.

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., and Wang, Y. (2017). Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

Marchuk, G. (2012). *Numerical methods in weather prediction*. Elsevier.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Moretti, F., Pizzuti, S., Panzieri, S., and Annunziato, M. (2015). Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling. *Neurocomputing*, 167:3–7.

Murphy, A. H. (1993). What is a good forecast? an essay on the nature of goodness in weather forecasting. *Weather and Forecasting*, 8(2):281–293.

Niaki, S. T. A. and Hoseinzade, S. (2013). Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, 9(1):1.

Nix, D. A. and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *International Conference on Neural Networks (ICNN)*, pages 55–60.

Osoba, O., Mitaim, S., and Kosko, B. (2013). The noisy expectation–maximization algorithm. *Fluctuation and Noise Letters*, 12(3):1350012.

Pan, Z., Liang, Y., Wang, W., Yu, Y., Zheng, Y., and Zhang, J. (2019). Urban traffic prediction from spatio-temporal data using deep meta learning. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1720–1730.

Pearce, T., Zaki, M., Brintrup, A., and Neely, A. (2018). High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *International Conference on Machine Learning (ICML)*, pages 4072–4081.

Perc, M. (2005). Nonlinear time series analysis of the human electrocardiogram. *European Journal of Physics*, 26(5):757.

Perc, M., Ozer, M., and Hojnik, J. (2019). Social and juristic challenges of artificial intelligence. *Palgrave Communications*, 5(1):1–7.

Persson, C., Bacher, P., Shiga, T., and Madsen, H. (2017). Multi-site solar power forecasting using gradient boosted regression trees. *Solar Energy*, 150:423–436.

Qi, X., Hou, K., Liu, T., Yu, Z., Hu, S., and Ou, W. (2021). From known to unknown: Knowledge-guided transformer for time-series sales forecasting in alibaba. *arXiv preprint arXiv:2109.08381*.

Qiu, M., Zhao, P., Zhang, K., Huang, J., Shi, X., Wang, X., and Chu, W. (2017). A short-term rainfall prediction model using multi-task convolutional neural networks. In *International Conference on Data Mining (ICDM)*, pages 395–404. IEEE.

Razzak, F., Yi, F., Yang, Y., and Xiong, H. (2019). An integrated multimodal attention-based approach for bank stress test prediction. In *International Conference on Data Mining (ICDM)*, pages 1282–1287. IEEE.

Ren, X., Li, X., Ren, K., Song, J., Xu, Z., Deng, K., and Wang, X. (2021). Deep learning-based weather prediction: a survey. *Big Data Research*, 23:100178.

Richardson, L. F. (2007). *Weather prediction by numerical process*. Cambridge University Press.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., and Aigrain, S. (2013). Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A*, 371(1984):20110550.

Rodrigues, F. and Pereira, F. C. (2020). Beyond expectation: deep joint mean and quantile regression for spatiotemporal problems. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: convolutional networks for biomedical image segmentation.

Rudin, C. (2019). Do simpler models exist and how can we find them? In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1–2.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Sandve, G. K., Nekrutenko, A., Taylor, J., and Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLoS Computational Biology*, 9(10):e1003285.

Sapankevych, N. I. and Sankar, R. (2009). Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine*, 4(2).

Shah, A., Kadam, E., Shah, H., Shinde, S., and Shingade, S. (2016). Deep residual networks with exponential linear unit. In *Proceedings of the International Symposium on Computer Vision and the Internet*, pages 59–65.

Sharma, N., Sharma, P., Irwin, D., and Shenoy, P. (2011). Predicting solar generation from weather forecasts using machine learning. In *International Conference on Smart Grid Communications (SmartGridComm)*, pages 528–533. IEEE.

Siddiqui, S. A., Mercier, D., Dengel, A., and Ahmed, S. (2020). Tsinsight: a local-global attribution framework for interpretability in time-series data.

Siddiqui, S. A., Mercier, D., Munir, M., Dengel, A., and Ahmed, S. (2019). Tsviz: demystification of deep learning models for time-series analysis. *IEEE Access*, 7:67027–67040.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85.

Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using LSTMs. In *International Conference on Machine Learning (ICML)*, pages 843–852.

Taieb, S. B., Yu, J., Barreto, M. N., and Rajagopal, R. (2017). Regularization in hierarchical time series forecasting with application to electricity smart meter data. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 4474–4480.

Targ, S., Almeida, D., and Lyman, K. (2016). Resnet in resnet: generalizing residual architectures. *arXiv preprint arXiv:1603.08029*.

Tolstykh, M. and Frolov, A. (2005). Some current problems in numerical weather prediction. *Izvestiya Atmospheric and Oceanic Physics*, 41(3):285–295.

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., and Troncoso, A. (2021). Deep learning for time series forecasting: a survey. *Big Data*, 9(1):3–21.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Voyant, C., Muselli, M., Paoli, C., and Nivet, M.-L. (2012). Numerical weather prediction (NWP) and hybrid ARMA/ANN model to predict global radiation. *Energy*, 39(1):341–355.

Wang, B., Lu, J., Yan, Z., Luo, H., Li, T., Zheng, Y., and Zhang, G. (2019a). Deep uncertainty quantification: A machine learning approach for weather forecasting. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2087–2095.

Wang, B., Yan, Z., Lu, J., Zhang, G., and Li, T. (2018a). Explore uncertainty in residual networks for crowds flow prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.

Wang, B., Yin, P., Bertozzi, A. L., Brantingham, P. J., Osher, S. J., and Xin, J. (2017a). Deep learning for real-time crime forecasting and its ternarization. *arXiv preprint arXiv:1711.08833*.

Wang, D., Zhang, J., Cao, W., Li, J., and Zheng, Y. (2018b). When will you arrive? Estimating travel time based on deep neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Wang, H. and Yeung, D.-Y. (2016). Towards Bayesian deep learning: a framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3395–3408.

Wang, J., Gu, Q., Wu, J., Liu, G., and Xiong, Z. (2016). Traffic speed prediction and congestion source exploration: A deep learning method. In *International Conference on Data Mining (ICDM)*, pages 499–508.

Wang, J., Lin, Y., Wu, J., Wang, Z., and Xiong, Z. (2017b). Coupling implicit and explicit knowledge for customer volume prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 1569–1575.

Wang, J., Zheng, V. W., Liu, Z., and Chang, K. C.-C. (2017c). Topological recurrent neural network for diffusion prediction. In *International Conference on Data Mining (ICDM)*, pages 475–484.

Wang, Q., Zhao, W., Yang, J., Wu, J., Hu, W., and Xing, Q. (2019b). Deeptrust: a deep user model of homophily effect for trust prediction. In *International Conference on Data Mining (ICDM)*, pages 618–627. IEEE.

Wasay, A., Hentschel, B., Liao, Y., Chen, S., and Idreos, S. (2018). Mothernets: rapid deep ensemble learning.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., and Sun, L. (2022). Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*.

Wikipedia contributors (2022). Time series — Wikipedia, the free encyclopedia. [Online; accessed 17-February-2022].

Wikiquote (2020). George e. p. box — wikiquote,.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430.

Wu, Y. and Tan, H. (2016). Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv preprint arXiv:1612.01022*.

Xiao, Y., Cai, J., Yang, Y., Zhao, H., and Shen, H. (2018). Prediction of microrna subcellular localization by using a sequence-to-sequence model. In *International Conference on Data Mining (ICDM)*, pages 1332–1337.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 802–810.

Yan, X., Zhang, W., Ma, L., Liu, W., and Wu, Q. (2018). Parsimonious quantile regression of financial asset tail dynamics via sequential learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1575–1585.

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P., Ye, J., and Li, Z. (2018). Deep multi-view spatial-temporal network for taxi demand prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 2588–2595.

Yao, W., Zeng, Z., and Lian, C. (2017). Generating probabilistic predictions using mean-variance estimation and echo state network. *Neurocomputing*, 219:536–547.

Ye, Y., Zheng, Y., Chen, Y., Feng, J., and Xie, X. (2009). Mining individual life pattern based on location history. In *International Conference on Mobile Data Management: Systems, Services and Middleware (MDM)*, pages 1–10.

Yeo, K., Melnyk, I., Nguyen, N., and Lee, E. K. (2018). DE-RNN: forecasting the probability density function of nonlinear time series. In *International Conference on Data Mining (ICDM)*, pages 697–706.

Yi, X., Zhang, J., Wang, Z., Li, T., and Zheng, Y. (2018). Deep distributed fusion network for air quality prediction. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 965–973.

Yi, X., Zheng, Y., Zhang, J., and Li, T. (2016). ST-MVL: Filling missing values in geo-sensory time series data. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2704–2710.

Yu, R., Li, Y., Shahabi, C., Demiryurek, U., and Liu, Y. (2017). Deep learning: A generic approach for extreme condition traffic forecasting. In *SIAM International Conference on Data Mining (SIAM)*, pages 777–785.

Yuan, Y., Xun, G., Ma, F., Wang, Y., Du, N., Jia, K., Su, L., and Zhang, A. (2018). Muvan: a multi-view attention network for multivariate temporal data. In *International Conference on Data Mining (ICMD)*, pages 717–726.

Zeng, X. and Lu, J. (2018). Decision support systems with uncertainties in big data environments. *Knowledge-Based Systems*, 143:327.

Zhang, J., Zheng, Y., and Qi, D. (2017a). Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 1655–1661.

Zhang, J., Zheng, Y., Qi, D., Li, R., and Yi, X. (2016). Dnn-based prediction model for spatio-temporal data. In *International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 1–4.

Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., and Li, T. (2018). Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artificial Intelligence*, 259:147–166.

Zhang, L., Aggarwal, C., and Qi, G.-J. (2017b). Stock price prediction via discovering multi-frequency trading patterns. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2141–2149.

Zhang, W., Quan, H., and Srinivasan, D. (2019). An improved quantile regression neural network for probabilistic load forecasting. *IEEE Transactions on Smart Grid*, 10(4):4425–4434.

Zhang, X., Qian, B., Li, Y., Yin, C., Wang, X., and Zheng, Q. (2019). Knowrisk: an interpretable knowledge-guided model for disease risk prediction. In *International Conference on Data Mining (ICDM)*, pages 1492–1497. IEEE.

Zhao, Y., Shen, Y., Zhu, Y., and Yao, J. (2018). Forecasting wavelet transformed time series with attentive neural networks. In *International Conference on Data Mining (ICDM)*, pages 1452–1457.

Zheng, Y., Capra, L., Wolfson, O., and Yang, H. (2014). Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3):38.

Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., and Gai, K. (2019). Deep interest evolution network for click-through rate prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 5941–5948.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. (2022). Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. *arXiv preprint arXiv:2201.12740*.

Zhou, Z. and Matteson, D. S. (2015). Predicting ambulance demand: A spatio-temporal kernel approach. In *International Conference on Knowledge Discovery and Data Mining*, pages 2297–2303. ACM.

Zhu, L. and Laptev, N. (2017). Deep and confident prediction for time series at uber. In *International Conference on Data Mining Workshops (ICDM)*, pages 103–110.

Ziat, A., Delasalles, E., Denoyer, L., and Gallinari, P. (2017). Spatio-temporal neural networks for space-time series forecasting and relations discovery. In *International Conference on Data Mining (ICDM)*, pages 705–714.

Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2017). Recurrent highway networks. In *International Conference on Machine Learning (ICML)*, pages 4189–4198.