UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Towards Adaptive, Scalable and Interpretable Spatial-Temporal Forecasting

by Jinliang Deng

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Professor Ivor W. Tsang, Associate Professor Xuan Song

University of Technology Sydney
Faculty of Engineering and Information Technology

August 2023

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Jinliang Deng declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in 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.

This document has not been submitted for qualifications at any other academic institution.

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

Signature:_____

Date: 13/12/2023

I would like to dedicate this thesis to my loving families ...

Acknowledgements

As I reflect upon the culmination of my PhD journey, the vivid memories of my first day at the University of Technology Sydney come flooding back. This journey, though seemingly brief, has been profound and transformative.

Foremost, I extend my heartfelt gratitude to my esteemed supervisor, Prof. Ivor W. Tsang. His unwavering support and guidance have been instrumental throughout my PhD journey. Prof. Tsang has provided me with the freedom to delve into intriguing research topics, always prioritizing my growth and development over mere outputs. His vast knowledge and strategic insights have consistently illuminated my research path. To me, he embodies the epitome of a scholar: knowledgeable, discerning, kind, and patient. The wisdom I have gleaned from him transcends academia, shaping my broader perspective on life. His influence on my academic journey is immeasurable, and I consider myself truly fortunate to have been mentored by him.

I am equally indebted to my co-supervisor, Associate Professor Xuan Song. Professor Song not only provided me with an outstanding research platform but also showed great concern for both my academic pursuits and daily life. I would like to convey my deepest appreciation for Professor Song's invaluable support and nurturing.

I also wish to extend my gratitude to Prof. Chengqi Zhang, Prof. Jie Lu, Prof. Xindong Wu, Prof. Hui Xiong, and Prof. Qianxiang Wang. Their support and insightful suggestions have significantly enriched my research and academic trajectory.

My PhD journey was made all the more enriching by the camaraderie and support of my colleagues and friends. I am profoundly grateful to Dr. Renhe Jiang, Dr. Bo Han, Dr. Yuangang Pan, Dr. Yaxin Shi, Dr. Yueming Lyu, Dr. Yan Zhang, Dr. Xiaofeng Xu, Dr. Xu Chen, Dr. Xiaowei Zhou, Dr. Xiaofeng Cao, Dr. Ziyue Qiao, Jing Li, Xingrui Yu, Yinghua Yao, Bowen Xing, Cheng Chen, Du Yin, Yongkang Li, Hangchen Liu, Hongbin Xie, Defan Feng, Yichen Zhao, Jiaqi Zhang, Bin Wang, Xiusi Chen, Yi Yang, Yiyi Luo and many others. Your companionship has made this journey memorable. A special mention to Dr. Xiaowei Zhou, Xiusi Chen, and Du Yin for standing by me during challenging times. To all those unmentioned, your presence in my PhD journey will always be cherished.

Lastly, but most importantly, I owe a debt of gratitude to my parents. Their unwavering support, understanding, and love have been my anchor. Completing this PhD would have been an insurmountable task without them. My love for them knows no bounds.

Abstract

Abstract: In recent years, the application of deep learning models in the realm of spatial-temporal forecasting has significantly outpaced traditional statistical learning methods such as ARMA, VAR, and STL. However, a notable drawback to these models is their 'black box' nature, leaving their internal workings largely uninterpretable. Furthermore, recent studies have exposed the limitations of deep learning models, particularly those within the Transformer family, which often fall short when compared to carefully crafted linear models in various situations. These revelations invite reconsideration of the indispensability of deep learning models in spatial-temporal forecasting.

This thesis seeks to explore the complexities of modeling spatial-temporal data, striving to devise an interpretable, adaptive, and scalable spatial-temporal forecasting model that can compete with current state-of-the-art methodologies. By accomplishing this, it is anticipated that a deeper understanding of the underlying processes will be achieved, potentially paving the way for advancements in the field of spatial-temporal forecasting. This research encapsulates three key contributions: 1. The development of an attention-based method to probe correlations within diverse spatial-temporal contexts, addressing both adaptability and interpretability challenges; 2. The invention of a normalization-based method to take advantage of the characteristics of spatial and temporal data distribution, grappling with

adaptability, interpretability, and scalability issues; 3. The introduction of a novel framework designed to emulate the dynamic behavior of underlying components in time series data, tackling adaptability, interpretability, and scalability concerns.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background and Research Objective

Time series forecasting has long been a pivotal area of inquiry within the disciplines of machine learning, statistics, and data science. Its applications are manifold, extending across a diverse array of sectors including, but not limited to, finance, healthcare, energy management, and telecommunications. While traditional univariate time series models such as Auto-Regressive Integrated Moving Average (ARIMA) and Exponential Smoothing techniques have demonstrated efficacy in isolated systems, they often fall short in capturing the complexity inherent in open and multi-variable systems. It is in this context that spatial-temporal forecasting emerges as a critical extension, allowing for the simultaneous analysis of multiple, interrelated temporal variables.

In the financial sector, spatial-temporal forecasting can be employed to model the intricate relationships between stock prices, trading volumes, and macroeconomic indicators. In healthcare, it can be used to predict patient outcomes based on a multitude of variables like vital signs, medication schedules, and other treatment parameters. The field also holds promise for climate science, where variables

such as temperature, humidity, and wind speed are intrinsically linked. Moreover, in the realm of supply chain management, spatial-temporal forecasting can offer invaluable insights by analyzing the interdependencies between inventory levels, demand fluctuations, and production schedules.

The overarching aim of this research is to advance the state-of-the-art in spatial-temporal forecasting by developing models that are not only powerful but also adaptive, scalable and interpretable. This involves a two-pronged approach: firstly, the extension and refinement of existing statistical methodologies to accommodate multivariate data; and secondly, the integration of cutting-edge machine learning algorithms such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Graph Neural Networks (GNNs) into the forecasting framework.

By pushing the boundaries of what is currently achievable in spatial-temporal forecasting, this research aspires to catalyze transformative changes across a multitude of applications. The ultimate objective is to provide actionable, accurate, and timely forecasts that can significantly improve decision-making processes and optimize resource allocation in complex, multi-variable systems.

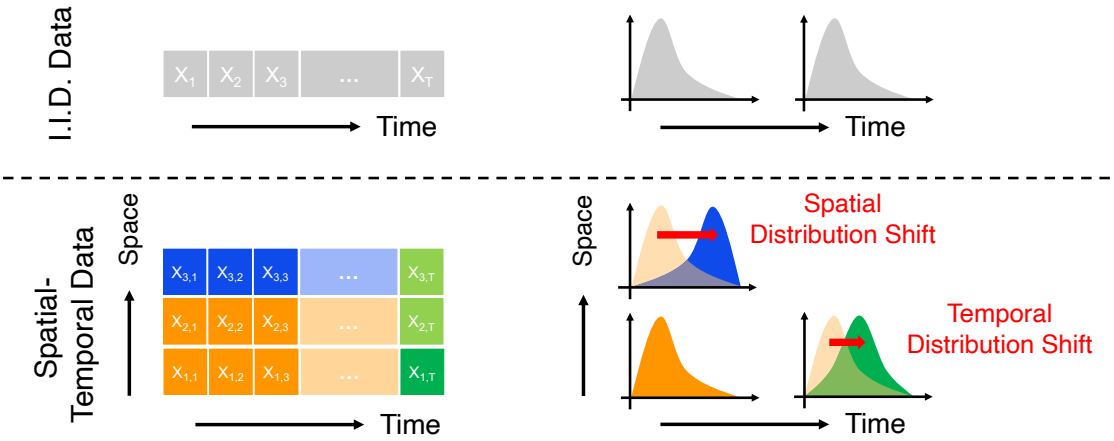
1.2 Research Questions and Motivations

As discussed in the last section, the research objective is to build interpretable, adaptive and scalable deep-learning models for spatial-temporal forecasting. This derives the two **RESEARCH QUESTIONS** of this thesis, namely *space-time-varying correlations and spatial-temporal scalability*, that set spatial-temporal forecasting apart from other machine learning tasks.

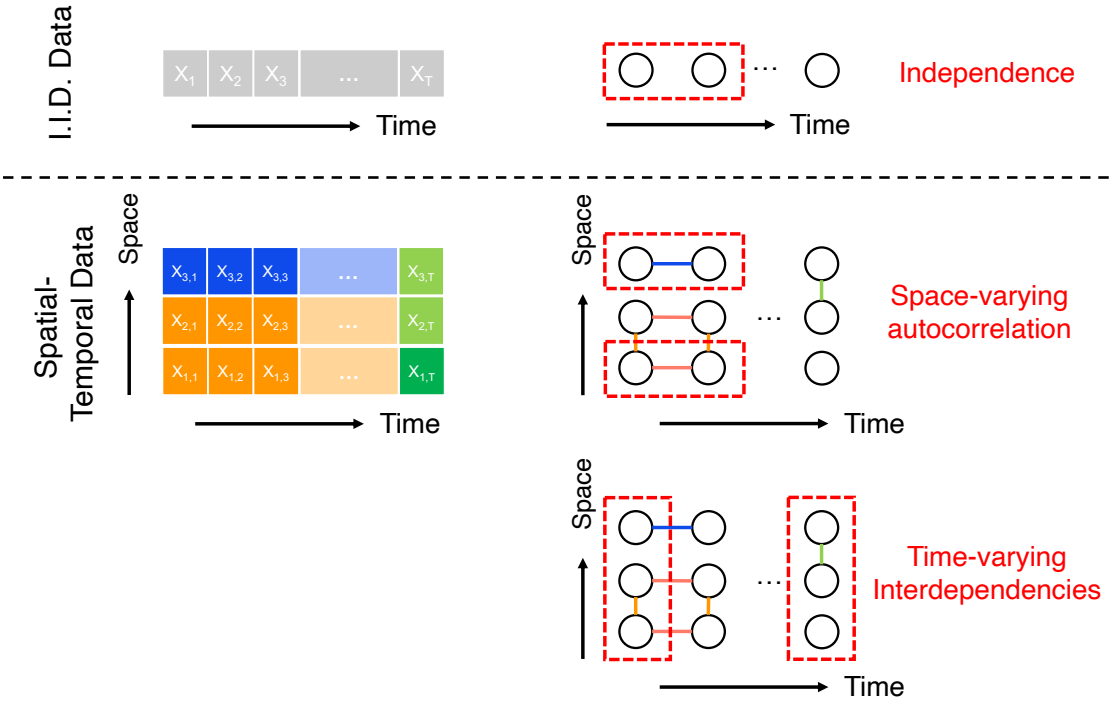
1.2.1 Space-time-varying Correlations

Clarification of variables: Before introducing space-time-varying correlation, we clarify the concept of variables in our context. Instead of complying with that in the context of MTS, which is defined as a sequence of chronologically ordered observations, we base the definition of the variable on space and time jointly. To be more specific, a spatial point paired up with a temporal point corresponds to a variable in our modeling. This definition appears to be counter-intuitive, given that each variable can only produce at most one observation, instead of obtaining an array of observations as in a common supervised learning scenario. Key support for the identification of the spatial-temporal points as variables rather than pure observations is that they tend to be not independently and identically distributed. Just as Heraclitus, the ancient Greek pre-Socratic philosopher, said, *"No man ever steps in the same river twice. For it's not the same river and he's not the same man."* This metaphor also works well for the dynamics of time series which are subject to constant changes, especially in the open environment. Therefore, the spatial-temporal points better fit the concept of variable than pure observation.

Space-time-varying correlations: One of the most salient challenges in spatial-temporal forecasting pertains to the dynamic nature of correlation patterns across both spatial and temporal dimensions. Traditional machine learning models often operate under the assumption of identically distributed and independent data samples over time. This assumption, however, is frequently violated in the realm of spatial-temporal data. Specifically, the data distribution is subject to shifts across different spatial locations and temporal intervals, as illustrated in Fig.1.1a, thereby undermining the foundational assumptions that ensure the efficacy of machine



(a) Spatial-temporal Distribution Shifts.



(b) Space-time-varying Inter-dependencies.

Figure. 1.1 Space-time-varying Correlations

learning and deep learning algorithms. Such distributional shifts can significantly compromise the performance of these models, particularly when they are applied to new, unseen data.

This issue is further exacerbated by the intricate interdependencies that characterize spatial-temporal data samples, as illustrated in Fig.1.1b. Unlike the assumption of independence, these data samples are often entangled in complex relationships that can fluctuate both temporally and across different series. Accurately capturing these interdependencies is not merely an added nuance; it is a critical requirement for generating reliable forecasts. The presence of these dynamic interdependencies introduces an additional layer of complexity that any robust spatial-temporal forecasting model must effectively address. Failure to account for these complexities can result in models that are not only less accurate but also less reliable when applied to real-world scenarios, thereby limiting their utility in decision-making processes.

Causes of space-time-varying correlations: This heterogeneity in correlations can be driven by numerous factors, such as alterations in environmental conditions, policy changes, or technological innovations, leading to differing shifts in the underlying data distributions across the time series. In addition, these factors trigger heterogeneous patterns in time series data, some of which persist for an extended time, some recur periodically, and some occur irregularly showing volatile fluctuations. Take the daily returns of different stocks in a stock market as a typical scenario, factors such as changes in market conditions, macroeconomic policies, or the introduction of disruptive technologies can lead to shifts in the correlations between different stocks. Specifically, two technology companies might exhibit a stronger correlation during a period of rapid technological advancement,

while the same companies might show a weaker correlation during an economic downturn that affects different sectors of the economy unevenly. As for another scenario of retail sales across multiple product categories, seasonal products like winter clothing may exhibit sales peaks during colder months, while non-seasonal products like kitchenware may have more evenly distributed sales throughout the year. Promotional events could also cause sudden shifts in sales distribution, but these shifts would likely impact different product categories to varying extents, depending on the nature of the promotion.

Relation with other relevant concepts: Instead of in line with the convention of this field to use *spatial-temporal correlation*, we coin *space-time-varying correlation* for the purpose of highlighting the changing behavior of statistical properties of time series data over space and time. Furthermore, we also find three related and commonly referred concepts in the relevant literature, consisting of *heteroscedasticity* [36, 15], *distribution shift* [48, 132] and *concept drift* [133, 64, 27]. In statistics, heteroscedasticity happens when the standard deviations of a predicted variable, monitored over different values of an independent variable or as related to prior time periods, are non-constant [36, 15]. In the field of machine learning and statistics, a distribution shift refers to a change in the way data is distributed. Concept drift and distribution shift are related but distinct terms in the machine learning literature, both pertaining to changes in the underlying data that a model is working with. A distribution shift refers to a change in the distribution of data [48, 132]. It's a broad term that encompasses any situation where the probability distribution of the data changes. This could be due to changes in the input features (covariate shift), changes in the output variable (prior probability shift), or changes in the relationship between input and output variables. Concept drift is a specific type

of distribution shift that occurs when the relationship between the input features (also known as predictors or covariates) and the output variable (also known as the target or response variable) changes over time [133, 64, 27]. In other words, it's when the function that maps inputs to outputs — the "concept" that the model is trying to learn — changes.

Challenges to conventional statistical learning methods: Space-time-varying correlations impose tremendous challenges to time series forecasting, carrying significant implications. Traditional statistical models, such as autoregressive integrated moving average (ARIMA) models [38] or vector autoregressive (VAR) models [40] work in a restrictive scenario where the variables are linearly interdependent and the correlations among the variables are constant over time. Extending the ability of these models to handle more complicated scenarios requires the practitioners to manually specify the form of the model that possesses the capacity to accurately characterize the dynamics of time series data, potentially by constructing various interactions among the variables based on their understanding of the data dynamics. However, the introduction of high-order interactions among the variables substantially complicates the estimation of the model's parameter, often disabling the analytical solution. Besides, this method still falls short in handling the space-time-varying correlations. To model varying correlations, intensive and complex computations are necessitated. This complexity arises due to the need to constantly update the correlation matrix as new data comes in, especially in a high-dimensional time series context where the number of correlations grows super-linearly with the number of variables. Each update requires a new estimation of the correlation matrix, and the computational cost can be substantial, particularly for large datasets. The use of more advanced models, like Dynamic Conditional

Correlation (DCC) models [22] or copula models [74], can help address this issue, but these models themselves are also complex and computationally demanding. They involve additional layers of estimation and often rely on iterative algorithms, which can be time-consuming to run, especially on large datasets. Furthermore, these models require careful specification and validation, adding to their complexity.

Challenges to modern deep learning methods: Advanced deep-learning techniques can capture the space-time-varying correlations to a certain extent, but they still struggle with the issue of distribution shifts, especially in those cases where the change occurs without explicit indications or much regularity [129, 117, 118]. The failure in handling distribution shifts is attributed not to the low capacity, but to the low generalization ability of the model. Theoretically, a multilayer perceptron (MLP) with a sufficient number of hidden units and hidden layers possesses the ability to mimic any dynamics. However, this success typically necessitates large volumes of data samples drawing from various patterns uniformly. For those patterns with limited or no data samples, the prediction accuracy provided by the model will substantially degrade. This requirement on the data is typically not met in a time series context, given that there are patterns that only appear at some points in the future and cannot be accessed when the model is trained. In addition, since the data distribution constantly changes, even for the patterns that are already present, we may only collect a restricted number of samples of them that are corrupted with noise. In this case, the optimizer is prone to be misled by noisy signals, incurring the overfitting issue. As showcased by [44], the dedicated handling of the trend, which serves as a measure of data distribution, can strengthen the generalization ability of the deep learning model on data that inherently behaves gradual shift in trend level.

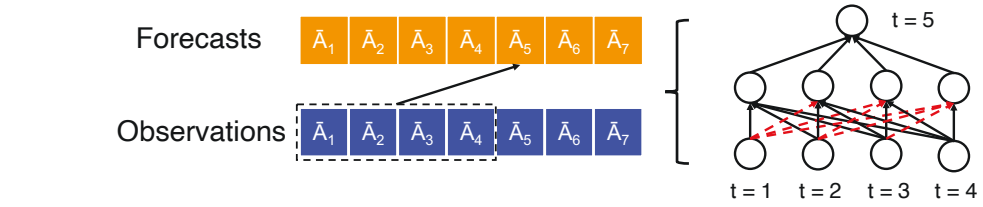
In summary, space-time-varying correlation in time series data poses a unique challenge in time series forecasting, demanding an intensive investigation of appropriate analytical strategies. Despite the associated challenges, addressing this issue effectively can substantially enhance the performance of forecasting models.

1.2.2 Spatial-temporal Scalability

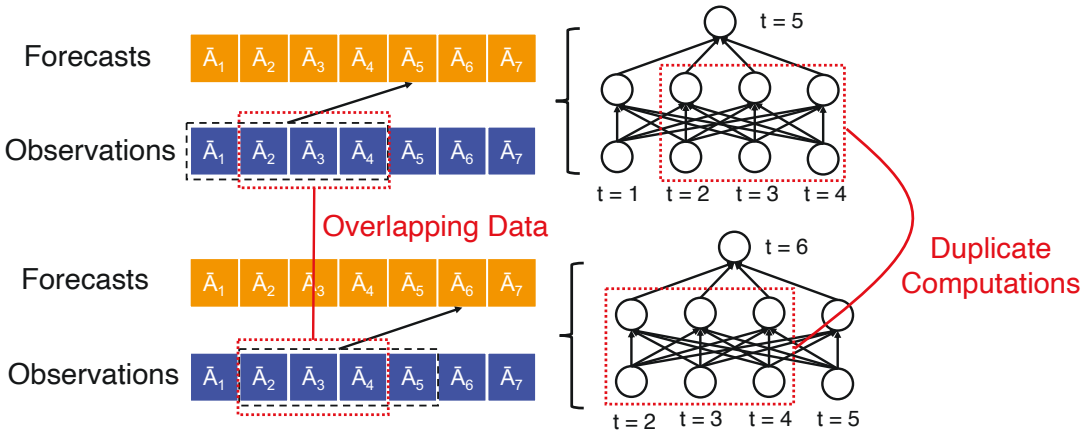
As the magnitude of data collection continues to expand, the management and processing of large-scale time series data have become increasingly complex tasks. This necessitates the development of scalable prediction algorithms that can efficiently handle vast datasets while maintaining high forecast accuracy.

The challenge of scalability becomes more complex due to the intricate long-term dependencies found in time series data. Forecasting typically involves using an extended sequence of historical data as input to make predictions about future values. In several instances, the size of the input sequence may span numerous consecutive days. Prediction accuracy generally improves as the size of the input sequence increases, however, at the cost of intensive computations, leading to longer latency in the prediction process. This becomes problematic in domains that have little tolerance for delay, like financial markets.

The implications of scalability: Scalability is a measure of an algorithm or system's capacity to manage an increasing workload, typically quantified by the volume of computations performed. If an algorithm can handle a growing volume of computations without significantly increasing computational resources or time, it's considered to have good computational scalability. The potential for parallelization — the capacity to execute multiple computations or processes concurrently — is also a vital factor influencing computational scalability. If an algorithm or system



(a) Unnecessary Computations.



(b) Redundant Computations.

Figure. 1.2 Spatial-temporal Scalability

can effectively leverage parallel computing resources (like multi-core processors or distributed computing systems), it can significantly improve its computational scalability. Therefore, by minimizing computations and promoting parallelization, an algorithm's scalability can be substantially improved.

Unnecessary computations: Traditional neural network architectures, such as Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Transformers, are designed with densely-connected and symmetric structures. These architectures allocate uniform computational resources, including the number of parameters and operations, to each observation in the input sequence. While this

design is well-suited for tasks like time series classification, it may not be the most efficient for forecasting tasks, which inherently possess a skewed structure.

In the realm of time series forecasting, more recent data points often carry greater predictive value and thus warrant more complex computational engagement for effective information extraction. Conversely, earlier observations may not require such computational intensity, as they often contribute less to the accuracy of future forecasts. This asymmetry in the importance of data points suggests that the traditional symmetric and densely connected neural network architectures may introduce unnecessary computational complexity, as illustrated in Fig. 1.2a. This is particularly true for earlier observations, which do not significantly contribute to the forecast and yet consume equal computational resources.

Redundant computations: The second challenge pertains to redundant computation, especially during the inference phase of spatial-temporal forecasting. Once deployed, forecasting models often operate on a rolling basis, which leads to overlapping data and, consequently, redundant computations between consecutive runs, as illustrated in Fig. 1.2b. This redundancy can be particularly costly in terms of computational resources, especially when dealing with extensive input sequences.

Such computational redundancy not only increases operational costs but also can slow down real-time forecasting applications where timely decision-making is crucial. Eliminating this redundancy is a focal point of our research. Our investigations aim to address these scalability challenges by developing more efficient computational strategies, thereby contributing to the advancement of spatial-temporal forecasting models that are both effective and resource-efficient.

One potential solution to eliminate redundant computations is to reuse computation results across executions, reducing the need to process the entire input sequence. If the features of the segment are extracted appropriately, they can remain useful in subsequent executions, thus eliminating repetitive and unnecessary computations.

Different deep architectures have different potentials to be optimized in terms of computational efficiency in the online testing mode. For instance, models such as the Multilayer Perceptron (MLP) [117, 118] and Transformer-based methods [129, 102] exhibit computational complexities that super-linearly correspond to the number of backward steps input to the model, irrespective of whether they are deployed in offline training or online testing mode. This is attributed to their intrinsic design to measure the contributions of each backward step to the forward horizon independently. Consequently, this prompts the need for continuous re-estimation of the contribution from every preceding step as time progresses. In contrast, Temporal Convolutional Networks (TCNs) [3] achieve computational complexity that is linearly proportional to the size of the input sequence in offline training mode, but sublinearly proportional in online testing mode, thanks to the reuse of features extracted in prior executions. Models within the Recurrent Neural Network (RNN) family [92, 91, 125] are more efficient, achieving minimal computation cost on a rolling basis due to their state variable that memorizes useful information propagated from the past. However, their performance tends to lag behind TCN and Transformer-based models. In essence, there is a noticeable vacuum in the availability of algorithms that are both effective and scalable.

Relation with space-time-varying correlation: The pursuit of developing scalable spatial-temporal inference algorithms is intrinsically linked with managing

space-time-varying correlations. Improving scalability inherently involves an efficient estimation of the shifting correlations in an online manner, which prompts us to investigate the structure and changing patterns of the correlations. Given that the correlations typically vary smoothly at a low rate, the calculation at the current iteration has great potential to be reused with minor adjustments at subsequent iterations. Furthermore, real-world time series data tend to exhibit multiple types of correlations with heterogeneous changing patterns, such as trend and seasonality. As a result, different correlations have to be decomposed and modeled individually.

1.2.3 Contrast with Other Tasks

To further underscore the distinction of time series forecasting, it is insightful to compare it with other common machine learning tasks. Specifically, tasks in the domains of image recognition and natural language processing, while complex in their own right, present different challenges and characteristics. As we will see in the following paragraphs, the dynamic and heterogeneous nature of time series data contrasts sharply with the more static and less distributionally volatile nature of data in image recognition and NLP tasks.

Image Recognition: Image recognition, a subfield of computer vision, is concerned with identifying and detecting objects or features in digital images [19]. The task typically involves processing static, two-dimensional arrays of pixel intensities. For example, in an image classification task, the goal might be to determine whether a given picture contains a cat or a dog. The model would identify relevant features, such as the shape of the animal, the texture of the fur, and the colors present in the image. These features are extracted from spatial

correlations between neighboring pixels and are usually static across different images. The correlations do not change over time as they would in a time series dataset.

A key point is that in image recognition, the correlations between different features (e.g., the shape of the ears, the color of the fur) are typically fixed and do not exhibit the kind of heterogenous correlations found in time series data. For instance, the relationship between ear shape and fur color in determining whether an image contains a cat or a dog is not expected to change over time or under different conditions.

Further, the distribution of these features is generally stationary. For example, the distribution of shapes, colors, and textures in a large dataset of cat and dog images is expected to remain relatively stable unless there's a significant change in the image-capturing process or in the types of images being included in the dataset. This is in stark contrast to time series data, where the underlying distribution can shift due to a multitude of factors, leading to heterogeneous distribution shifts.

In summary, while image recognition involves complex patterns and structures, the nature of these patterns is fundamentally different from those encountered in time series data. The lack of temporal dynamics, evolving correlations, and distribution shifts distinguishes image recognition tasks from time series forecasting tasks, and models that work well for image recognition might not be suited to the unique challenges of time series forecasting.

Natural Language Processing (NLP): NLP is a subfield of artificial intelligence that focuses on the interaction between computers and humans through natural language. The ultimate objective of NLP is to read, decipher, understand, and make sense of human language in a valuable way.

Consider the task of sentiment analysis, where the goal is to determine the sentiment (e.g., positive, negative, neutral) of a given text. In this case, the sequence of words (discrete symbols) in a sentence or paragraph forms the input data. The model needs to identify relevant features, such as the presence of positive or negative words and the overall context, to determine the sentiment. The correlations between these features (e.g., the sequence and context of words) are primarily determined by the grammatical and semantic rules of the language. However, unlike in time series data, these correlations are generally static within a given language. For instance, in English, the phrase "not good" typically has a negative sentiment, regardless of when or where it is used. This static nature of correlations contrasts with the dynamic, heterogenous correlations observed in time series data, where the relationship between variables can change over time due to a variety of factors.

Regarding data distributions, while certain shifts can occur in NLP (for example, the increased usage of certain words or phrases over time), they are often less pronounced and less impactful on model performance compared to the heterogenous distribution shifts in time series data. In NLP tasks, the distribution shifts are usually related to changes in language use over time or across different domains, but the fundamental grammatical and semantic structures remain relatively stable.

In summary, while NLP tasks involve complex sequences and structures, the nature of these sequences is fundamentally different from those in time series data. The static correlations and less impactful distribution shifts distinguish NLP tasks from time series forecasting tasks, emphasizing the need for specialized models that can handle the unique challenges of time series data.

1.3 Contributions of the Thesis

This thesis presents potential solutions for the development of adaptive and scalable time-series forecasting models, as illustrated in Fig. 1.3. The key contributions of this thesis are outlined below:

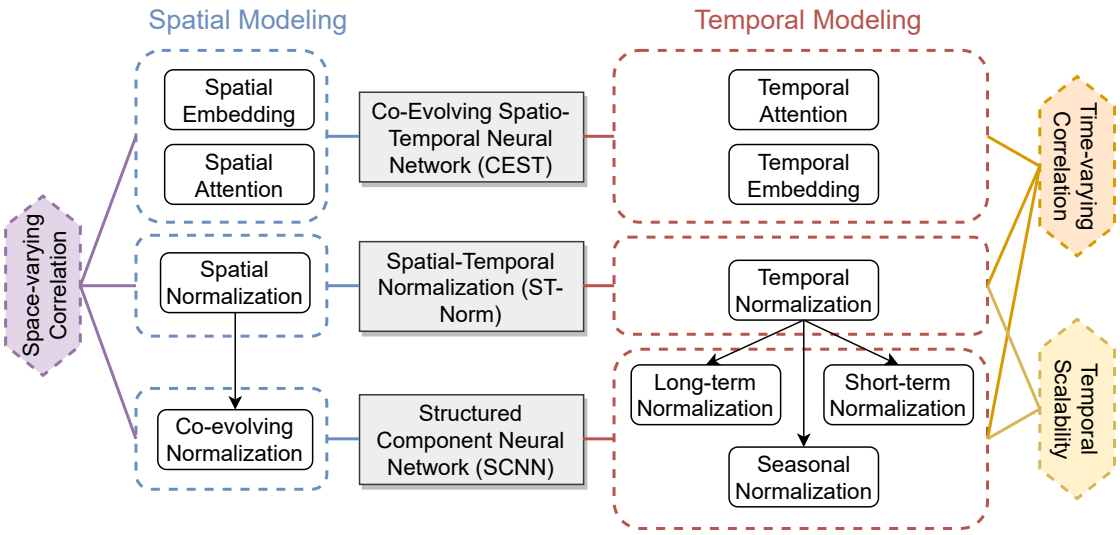


Figure. 1.3 Framework overview of three proposed methods. For each method, the techniques used for spatial and temporal modeling are labeled, along with the abilities demonstrated by these techniques. Arrows indicate the development of one technique from another, with the head side technique evolving from the tail side technique.

Contribution 1. We designed an attention-based method to investigate correlations within diverse spatial-temporal contexts, effectively addressing the issue of space-time-varying correlations.

- A multi-view representation learning module for time series data is proposed, integrating both periodic and real-time views.

- A co-evolving pattern learning and fusion module is introduced. This module extracts co-evolving patterns by identifying dynamic correlations between different time series.
- We implement multi-scale information-based forecasting, where the original time series represents fine-scale information and the co-evolving patterns represent coarse-scale information.

Contribution 2. We proposed a normalization-based method to extract the characteristics of spatial and temporal data distribution, effectively addressing both the space-time-varying correlations and spatial-temporal scalability issues.

- We provide both theoretical and empirical evidence that certain distribution-level features pose inherent challenges for deep learning models.
- Two types of normalization modules are proposed to refine the components representing spatial and temporal data distribution respectively.
- Extensive experiments are conducted to highlight the significant advantages of normalization-based methods in terms of efficacy, computational cost, and convergence speed.

Contribution 3. We proposed a novel framework aimed at simulating the changing behavior of the underlying distribution in time series data, effectively addressing both the space-time-varying correlations and spatial-temporal scalability issues.

- We introduce the Structured Component Neural Network (SCNN) for multivariate time series forecasting, marking the first fully decomposition-based neural architecture.

- A novel structural regularization method is proposed to explicitly shape the structure of the representation space learned from SCNN.
- Extensive experiments on three public datasets validate the effectiveness of SCNN, with observed general improvement over competing methods. Both empirical and analytical evidence demonstrate SCNN's superior performance in handling distribution shifts and anomalies, along with a significant reduction in computational cost.

1.4 Thesis Outline

The structure of the remaining chapters of this thesis is organized as follows.

- Chapter 2 summarizes the literature on time series forecasting, particularly the deep learning-based methods.
- Chapter 3 presents an attention-based method to investigate correlations within diverse spatial-temporal contexts.
- Chapter 4 presents a normalization-based method to extract the characteristics of spatial and temporal data distribution.
- Chapter 5 presents a novel framework aimed at simulating the changing behavior of underlying distributions in time series data.
- Chapter 6 summarizes the findings of this thesis and points out the directions of future work.

1.5 Publications

- **Jinliang Deng**, Xiusi Chen, Renhe Jiang, Xuan Song, and Ivor W. Tsang. "St-norm: Spatial and temporal normalization for multi-variate time series forecasting." In Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, pp. 269-278. 2021.
- **Jinliang Deng**, Xiusi Chen, Zipei Fan, Renhe Jiang, Xuan Song, and Ivor W. Tsang. "The pulse of urban transport: exploring the co-evolving pattern for spatio-temporal forecasting." ACM Transactions on Knowledge Discovery from Data (TKDD) 15, no. 6 (2021): 1-25.
- **Jinliang Deng**, Xiusi Chen, Renhe Jiang, Xuan Song, and Ivor W. Tsang. "A Multi-View Multi-Task Learning Framework for Multi-Variate Time Series Forecasting." IEEE Transactions on Knowledge and Data Engineering (2022).
- Jiewen Deng*, **Jinliang Deng***, Du Yin, Renhe Jiang, and Xuan Song. "TTS-Norm: Forecasting Tensor Time Series via Multi-way Normalization." ACM Transactions on Knowledge Discovery from Data (TKDD) (2023).
- **Jinliang Deng**, Xiusi Chen, Renhe Jiang, Du Yin, Yi Yang, Xuan Song, and Ivor W. Tsang. "Learning Structured Components: Towards Modular and Interpretable Multivariate Time Series Forecasting." IEEE Transactions on Knowledge and Data Engineering (2023). Under review.
- Renhe Jiang, Du Yin, Zhaonan Wang, Yizhuo Wang, Jiewen Deng, Hangchen Liu, Zekun Cai, **Jinliang Deng**, Xuan Song, and Ryosuke Shibasaki. "Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction." In Proceedings of the 30th ACM international conference on information & knowledge management, pp. 4515-4525. 2021.
- Yongkang Li, Zipei Fan, Du Yin, Renhe Jiang, **Jinliang Deng**, and Xuan Song. "HMGCL: Heterogeneous multigraph contrastive learning for LBSN friend recommendation." World Wide Web (2022): 1-24.
- Yongkang Li, Zipei Fan, Jixiao Zhang, Dengheng Shi, Tianqi Xu, Du Yin, **Jinliang Deng**, and Xuan Song. "Heterogeneous Hypergraph Neural Network for Friend Recommendation with Human Mobility." In Proceedings of the 31st

ACM International Conference on Information & Knowledge Management, pp. 4209-4213. 2022.

- Xiusi Chen, Yu Zhang, **Jinliang Deng**, Jyun-Yu Jiang, and Wei Wang. "Gotta: Generative Few-shot Question Answering by Prompt-based Cloze Data Augmentation." In Proceedings of the 2023 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics, 2023.
- Jiewen Deng, **Jinliang Deng**, Renhe Jiang, and Xuan Song. "Learning Gaussian Mixture Representations for Tensor Time Series Forecasting." The 32nd International Joint Conference on Artificial Intelligence (IJCAI'23).

Chapter 2

Literature Review

Deep learning models have achieved unparalleled success in computer vision. This success has spurred a vast body of literature related to deep learning methods, particularly in multivariate time series forecasting. Most research endeavors focus on drawing inspiration from computer vision, natural language processing, and graph data analysis, with only slight modifications or improvements to address the distinctive properties of spatial-temporal data. The rationale behind such cross-disciplinary applications is the intrinsic ability to frame spatial-temporal data as grids, graphs, sequences, or a combination of these, reflecting the abstract traits found in images, graphs, and languages.

In the realm of computational models, a majority of research relies on three core operations: the attention operator, the convolution operator, and the recurrent operator. Specifically, the attention operator has evolved into specialized forms such as spatial attention [25, 127, 58], temporal attention [131, 127], and sparse attention for computational efficiency [102, 129, 55]. Similarly, the convolution operator has been extended into spatial convolution [113, 56, 30], temporal convolution [104, 103], spatial-temporal convolution [33, 108], and adaptive convolution, which allows the operator's parameters to adapt to external conditions [71]. Lastly, the recurrent

operator has led to the development of gated recurrent units (GRU) [126], long short-term memory (LSTM) [110, 111], and adaptive RNNs [93, 78, 75, 71].

Despite notable advancements, a breakthrough solution with capabilities akin to GPT has yet to emerge. Recent research indicates that even a basic linear model, when augmented with trend decomposition techniques, can rival or surpass the performance of intricate deep learning models, particularly those based on the Transformer architecture. These findings have reverberated throughout the scientific community, challenging the prevailing belief that deep learning is revolutionizing time series forecasting as well. The limitations of deep learning models, especially those using the Transformer architecture, are often attributed to the inherent lack of structure in time series data. Unlike natural language, which consists of discrete tokens, time series data is made up of continuous points sampled at discrete intervals. This leads to a situation where two time series with diverging future trends may still be closely located in the feature space, making them hard to distinguish. To overcome this challenge, much of the existing literature aims to improve time series forecasting by incorporating the inherent structure of the time series as prior knowledge.

In our literature review, we conduct a comprehensive survey of this field, organizing studies based on the underlying framework they employ, such as the Transformer or MLP, and the specific type of structure they aim to enhance in time series data. This structure can be examined from two perspectives: temporal and spatial. The temporal perspective includes long-term, seasonal, and short-term components. These elements help to understand the time-based patterns and fluctuations in the data. On the other hand, the spatial perspective focuses on co-evolving and local components. The local component pertains to interconnected nodes within a

physical space. For example, in the context of traffic, intersections connected by the same road may show similar variations. The co-evolving component, however, affects nodes that are geographically dispersed but exhibit similar trends or behaviors. By categorizing the studies in this manner, we aim to provide a structured overview of the current state of research, highlighting the various approaches taken to address the complexities of time series forecasting.

2.1 Multi-layer Perceptrons

Zhang et al. [118] pioneered the study of utilizing deep learning models for spatial-temporal forecasting, demonstrating the potential of this burgeoning algorithmic paradigm in time series forecasting tasks. They developed a model called DeepST, derived from deep convolutional neural networks (CNNs) and tailored for handling grid-based spatial-temporal data, such as traffic flow. DeepST exhibits exceptional proficiency in projecting traffic flow, significantly outperforming traditional time series forecasting models, including ARIMA, SARIMA, and VAR. These models, while effective in certain scenarios, are limited in their ability to capture complex spatial-temporal dependencies. In a novel approach, Zhang et al. [118] formulated the snapshot of traffic flows distributed across an entire city as a channel of an image, represented by a matrix of pixels, in the sense that spatially adjacent values are correlated in the context of both spatial-temporal data and image data. This innovative representation allowed DeepST to take advantage of CNNs' capability to learn and recognize patterns in spatial data.

In the subsequent works [117, 119], they integrated the residual component into DeepST, resulting in an architecture called ST-ResNet, which resembles the well-known ResNet (Residual Network). The residual component further enhanced

DeepST's performance by facilitating the training process of model parameters, allowing them to converge at a more optimal position. This is achieved by adding shortcuts or skip connections between layers, enabling the network to learn more efficiently and effectively. Since ST-ResNet's inception, it has gained widespread popularity and drawn increasing attention to the field of deep learning-based spatial-temporal forecasting. It also serves as a widely adopted and competitive benchmark model in subsequent studies, setting a new standard for research in this area. This line of literature also includes [60, 26].

2.1.1 Short-term Enhanced MLP

By design, a Multilayer Perceptron (MLP) does not inherently capture the temporal order among data points in an input sequence, lacking what is known as 'inductive bias' for time series. To illustrate, consider an example where we permute the elements of any given input sequence in a fixed order. For any MLP, the permuted sequence would be treated as equivalent to the original sequence across all sample instances. This is because the weights of the MLP can be rearranged without the need for retraining, producing a consistent output for the permuted sequence that equals the original prediction. However, it is clear that such a permutation destroys the temporal structure inherent to the sequence. Therefore, this underscores the need to incorporate inductive biases to capture temporal relationships. First and foremost is to make the model aware of the neighboring relationship among the data points. One line of research employs convolution or moving average techniques to extract the short-term components [41, 90], which reflect the local trend over a short segment of observations. TimesNet [101] further extends this line of research by considering variations at multiple scales or resolutions.

2.1.2 Long-term (Trend) Ehanced MLP

The emergence of DLinear[116] challenges the need for using the Transformer as the primary model for time series forecasting, bringing the attention of the community back to exploring the ability of MLP enhanced by the components designed for recovering and modeling the structure of time series data. Zeng et al. [116] demonstrated that by employing a divide-and-conquer strategy, which separately models the evolution trend and the residual components with a straightforward linear model, the performance can compete against that of intricate Transformer-based models.

2.1.3 Seasonality Enhanced MLP

N-BEATS[68] offers a hierarchical decomposition approach that alternately isolates the trend and seasonal components of time series data, layer by layer. To endow the model with interpretability, the hypothesis space for trend components is restricted to polynomial functions of limited orders. Meanwhile, the seasonal components are modeled using the Fourier series. The coefficients associated with the polynomial functions and the Fourier series are learned from data via MLPs. N-HITS[8] is the follow-up work built upon N-BEATS.

DEPTS[24] estimates the seasonal component using a time-dependent function, specifically a series of cosine functions, independent of real-time data. This approach contrasts with existing methods, which rely on data collected over multiple previous periods to capture the seasonal component. One potential advantage of DEPTS lies in its reduced storage and computational demands, as well as its resilience to data noise, given that it solely uses the time variable to infer the seasonal component. However, this method has a limitation: the seasonal component will exhibit a

consistent pattern over time due to the periodicity of the cosine functions, limiting its ability to adapt to changes in the seasonal component. Other relevant studies include [99].

2.1.4 Position Encoding Enhanced MLP

Building on DLinear's foundation, TiDE [16] (an acronym for Time-series Dense Encoder) further integrates both dynamic covariates and static attributes into the time series data input. An MLP implements the model, enabling the capture of non-linear transitions between time steps. By including covariates, such as times of day, the model embeds the time series data within a structured feature space. Specifically, it biases the encoding to favor time series with identical or overlapping sequences of covariates, ensuring they are represented similarly, while minimizing correlations for time series with distinct or minimally shared covariates. A similar principle applies to static attributes. Consequently, the resulting feature space structure from this method is more robust than one without any structure prior.

Both ST-MLP [96] and STID [81] build upon this research direction, delving into the spatial-temporal domain. Their architectures are augmented by integrating both spatial and temporal embeddings. Moreover, ST-MLP [96] manages to encode a predefined graph structure as a set of biases specific to each series. This approach enables the physical distance to condition the structure of the feature space.

2.2 Recurrent Neural Networks

Despite the success of MLPs, they fall short in accounting for sequential order among input values, resulting in a limited ability to capture temporal correlations. To address this limitation, a series of studies [78] turned to the RNN (Recurrent

Neural Network) [76] framework, which inherently models sequential orders by recursively processing each observation in the sequence from beginning to end using the same pair of parameters. Moreover, the gating mechanism implemented by the sigmoid function in RNNs, such as the Long Short-Term Memory (LSTM) [39] and Gated Recurrent Unit (GRU) [13] networks, allows them to mimic the decaying behavior exhibited by the autocorrelation function, ensuring that observations in the distant past have limited or no impact on predictions. This adaptive memory management helps in effectively capturing long-term dependencies in time series data. These two distinctive and logical characteristics of RNNs have led to a surge in their application to spatial-temporal problems, addressing the limitations of earlier models and paving the way for more accurate and efficient forecasting in various domains. Representative works following this line include ConvLSTM [82], PredRNN [94, 92, 91], DeepAR[77].

2.2.1 Spatially-Enhanced RNN

The central issue in spatially-enhanced RNN is how to effectively model the inter-dependencies between different time series. These inter-dependencies can be sourced in two ways: Expert Input or Side Information: This includes information like road networks. Methods based on this source are exemplified by works like [110]. Data-Driven Approaches: These derive inter-dependencies through statistical analysis or coefficient estimation. Notable works in this category include [72]. Shang et al. [80] focus on learning the graph structure based on the complete time series sequences, rather than just the sub-sequences immediately preceding the forecast period. This results in an invariant graph structure that is resilient to sudden disruptions. However, this approach may not be suitable when the inter-dependencies change

over time. In contrast, BAI et al. [2] introduces a model that incorporates an adjacency matrix with learnable elements into the RNN framework. This allows the model to progressively update the inter-dependencies through backpropagation. Given that this model also assumes a consistent graph structure, it suffers from the same problem of performance degradation if the relationships between the time series change over time.

2.2.2 Seasonality Enhanced RNN

While RNNs can capture long-term dependencies through selective memory updates, they still inevitably experience significant information loss due to the gating effect, which grows exponentially over time. For instance, with a decay rate of 0.01 per hour, only about 20% of the information from the same time last week would be considered at the current time point. This limitation, arising from the recursive propagation of information, drives the optimization of RNN structures, such as introducing direct connections between non-consecutive time points. This allows crucial information from the distant past to be instantly transmitted to the present, bypassing cascading gates. However, this comes with a substantial increase in computational complexity if every pair of non-consecutive time points is explicitly connected.

To manage computational costs, existing studies leverage prior knowledge to select necessary links. For instance, some studies bridge periodic observations, allowing periodic information to pass through time steps without loss [134, 128]. These studies represent links as binary digits, where 1 signifies a connection, and 0 indicates its absence. Algorithmically, if links are sequentially connected, another RNN can be used to process the information held within the chain of connections.

This additional RNN, operating at specific intervals, can interact with the original RNN, which operates at each time step, to mutually enhance their capabilities. This approach can also be seen as a hierarchical multi-scale RNN, where multiple RNNs process time series data at different sampling rates in parallel.

However, this solution's drawback lies in its dependence on prior knowledge of recurring time series patterns, which limits its applicability when identifying such patterns is challenging. The emergence of the attention mechanism addresses this issue by learning connections without additional hints or supervision. Instead of using undifferentiable binary representations for connections, the attention mechanism employs a continuous number, known as the attention score, ranging from 0 to 1 to represent connection intensity. This enables optimization through gradient descent. Specifically, an attention score close to 0 indicates weak connections between associated observations, with little valuable information transfer, while a score near 1 suggests a strong correlation. Moreover, the values corresponding to an observation's historical connections are constrained to sum to 1. The desired outcome is that attention scores emphasize essential observations in history while suppressing irrelevant ones. Numerous studies fall into this category. Other studies [4] supplement RNN with decomposition techniques, forcing the RNN to optimize the prediction of the remainder component that cannot be explained by the seasonal component.

Recent studies seek to apply the Transformer framework, which comprises multiple layers of attention mechanisms and fully connected operations, to time series forecasting tasks. These efforts explore the possibility of discarding the recurrent architecture altogether.

2.3 Transformer Framework

The Transformer framework [88], which has its roots in natural language processing, has dominated the field of NLP due to the unprecedented success of models like BERT [20], BART [51], GPT [5], and, recently, ChatGPT [69]. Both natural language and time series data share the characteristic of exhibiting sequential patterns, making the Transformer an appealing choice for time series forecasting. As a result, it has served as the backbone architecture in numerous studies. However, the symmetric and parallel architecture of the Transformer makes it less suited for processing time series data compared to RNNs with sequential architectures, where RNNs inherently perceive the temporal order of observations based on the order in which they are fed into the model. To imbue the Transformer with the capability to capture sequential orders, researchers explicitly encode the positions of tokens into a vector space. The intricate combination of positional encodings and token encodings gives rise to the ability to capture rich and abundant semantics. Consequently, positional encodings are of fundamental importance to the Transformer's success.

In the context of time series forecasting, the design of positional encodings takes center stage, becoming even more critical than in the NLP context. This is because spatial-temporal data exhibit more complex structures than natural language, including both sequential structure, seasonal structure, and spatial structure. Positional encodings significantly determine the type of temporal structure the Transformer can perceive in addition to the input data. For instance, some studies draw inspiration from NLP and implement positional encodings using sine and cosine functions of their indexes in the sequence. Later studies found that enabling the positional encodings to be learnable, instead of freezing them, can

further strengthen the Transformer, significantly boosting the forecast accuracy. From then on, much attention has been brought to the Transformer framework.

Despite widespread applications, the Transformer framework does not demonstrate a dominant advantage over other frameworks in the field of time series forecasting. In some cases, even a simple linear model [115] or an MLP [16] can outperform Transformer-based state-of-the-art models, if they also incorporate the crafted position encodings. Therefore, whether the attention mechanism plays an indispensable role in time series forecasting, as it achieves in NLP, remains an open question.

Another concern is the prohibitive computational cost in training and making inferences with Transformers, as it asks for computing dot products between every pair of historical observations. A milestone study [129] showed that the attention scores follow a long-tailed distribution with a substantial amount of attention scores near zero. This implies that massive computations executed by Transformers are essentially useless in the scenario of time series forecasting. Therefore, improving the efficiency of the Transformer has become a hot subject in this field, giving rise to a series of insightful and pragmatic works.

2.3.1 Long-term (Trend) Enhanced Transformer

The fluctuations in long-term (trend) components over time introduce distribution shift issues, which can compromise the performance of Transformer models. RevIN [45] pioneered the approach to address this shift seen in the long-term component. It encases the Transformer or other core models with a normalization module at the initial layer and a denormalization module at the terminal layer. Specifically, the normalization module adjusts the mean and standard deviation of the input time

series data to 0 and 1, respectively. Reversely, the denormalization module brings the Transformer's projections back to the original mean and standard deviation.

The Non-stationary Transformer [63] builds on RevIN's framework, aiming to recognize temporal dependencies across data samples with varying measures of mean and standard deviation. This recognition stems from the understanding that normalization might mask dependencies associated with these statistics. Extending the approach of RevIN, it weaves the original statistics and the raw data into the calculation of attention scores, in conjunction with the queries and keys derived from the normalized data.

Fan et al. [23] noted that distribution shifts appear both across varying sampling frequencies and between the lookback window and the horizon window. From their viewpoint, this implies that traditional statistical methods might fall short in accurately measuring the genuine statistics of the distribution, which includes mean and standard deviation. While earlier methods relied on fixed long-term statistics to normalize time series data[45], they often struggled to navigate the complexities highlighted above. Addressing this challenge, the researchers introduced a neural module termed Dish-TS [23]. Dish-TS deploys an MLP to learn a level coefficient and a scaling coefficient, using backward data as its input. These coefficients are intended to offer a more nuanced depiction of the data distribution compared to the traditional mean and standard deviation, thereby addressing the intricate distribution shift challenge. To regulate these coefficient estimates, an auxiliary loss was incorporated, aiming to narrow the gap between the level coefficient and the traditional mean. In prior work, DAIN[73] also substantiated the importance of normalizing time series data with adaptive mean and standard deviation. The standout feature of Dish-TS and DAIN lies in its nuanced understanding of the

complexities associated with varying data distributions over time and across different sampling frequencies. Nonetheless, it overlooks the fact that when focusing on either the lookback or horizon window, the data might originate from a multimodal distribution. Such a distribution might be a poor fit for Gaussian approximation. Consequently, it's still up for debate as to what the coefficients generated by Dish-TS truly represent.

2.3.2 Seasonality Enhanced Transformers

Autoformer [102] is one of the earliest studies to highlight the limitations of traditional attention models in capturing seasonal components. It posits that spikes occurring at individual points in time series can mislead attention models, leading them to mistakenly assign high attention scores to false points [122]. To improve the accuracy of identifying time points with similar seasonal effects, the authors introduced a period-based attention mechanism. Instead of using dot-product similarities at individual time points, this mechanism employs autocorrelations at the period level for attention scores.

FEDformer [130], recognizing the inherent separation of seasonal and irregular components when representing time series using Fourier bases, shifts the representation of time series from the temporal to the frequency domain using a Fourier transform. In this domain, frequency modes with large amplitudes correspond to seasonal components, while those with smaller amplitudes relate to irregular components. Capitalizing on this observation, FEDformer reduces irregular noise by randomly omitting certain frequency modes, ensuring a small loss of seasonal information. Empirical evidence provided by the authors suggests that randomly selecting a mix of low and high-frequency modes preserves time series information,

predominantly consisting of seasonal components, more effectively than retaining only a fixed set of low-frequency components. However, it remains to be explored whether this random selection strategy outperforms a method focused solely on high-amplitude frequency components, as the latter might minimize information loss.

TDformer [122] shows that detrending time series data prior to Fourier transformation can maximize the advantages of attention models in the Fourier domain. Additionally, the study illustrates the difficulties attention models face in generalizing and extrapolating trend components, emphasizing the need for a specialized model for these components. In TDformer's approach, this is achieved by combining multiple average filters of varying sizes with an MLP for prediction.

Yet, a common oversight in these studies is the lack of a clear rationale for applying attention to frequency domain representations across different frequency modes. Since Fourier bases are mutually orthogonal and convey independent information, the benefits of inter-base information exchange remain unclear, warranting further exploration. While there are aspects of these methods that invite debate, their collective contributions to the field are undeniable. They collectively underscore that when seasonal components are processed effectively, there can be significant performance enhancements.

2.3.3 Spatially and Short-Term Enhanced Transformers

Much of the prior research has predominantly centered on bolstering the trend and seasonality components, with comparatively less emphasis placed on the spatial and short-term components. In our investigation, we identified Crossformer [123] as the pioneering work in this research direction. Subsequent advancements have

been made by DSFormer [114] and Scaleformer [79], which extend the capabilities of Crossformer by focusing on capturing multi-scale components.

Crossformer [123] enriches the information in the representation of time series by encoding data at the segment level and tapping into inter-dependencies between different series. This adaptation allows the Transformer-based model to identify short-term variations across the series, achieving marked advancements over earlier methods within the Transformer family [129, 102, 62]. We emphasize that the novelty of these techniques is apparent only when considered within the scope of methods that employ the Transformer framework. This is because similar concepts have already been explored in a range of pertinent studies related to traffic forecasting and multivariate time series forecasting [103, 104, 43], the earliest among which came up in 2019, four years prior to the development of Crossformer. The notable efficacy exhibited by Crossformer has brought the researchers' focus to the Transformers' constraints in harnessing short-term and spatial correlations. CARD [107] introduced a learnable token at the beginning of the input sequence to infuse statistical features characterizing the entire history of the relevant series. Furthermore, they employed the Exponential Moving Average (EMA) when calculating attention scores, giving more emphasis to short-term features—a strategy also embraced by Ma et al. [65], Woo et al. [100]. Building on this research trajectory, SageFormer[124] and PETformer[59] recently introduced distinct attention mechanisms to facilitate information propagation across series/channels.

While Crossformer has achieved notable success, it might not fully harness the potential of building interactions among the series. Some studies, such as PatchTST[67] (short for channel-independence patch time series Transformer) and CI[35] (short for Channel Independence training strategy), have exhibited

advantages on some datasets when applying the Transformer independently to each channel, which in our context is referred to as a series. These approaches contrast not only with conventional methods [129, 130, 102] that merge all channels and use this combined input for the Transformer but also with Crossformer [123], which takes into account the inter-dependencies among series. The authors of PatchTST further suggest that capturing local semantic information, in their case through a convolution operation with strides, can enhance Transformers that by nature operate on a single time step. As a result, there remains an open question regarding under what conditions spatial components are beneficial and when they might produce adverse effects.

While the intuitions behind these strategies have deep-seated roots, with some even finding similar implementations in traffic prediction and multivariate time series forecasting [17, 81], it's encouraging to observe scholars in the Transformer community catching up with advancements in related fields. We remain hopeful that groundbreaking and truly transformative research will further elevate the field of time series forecasting in the imminent future.

2.4 Graph Neural Network

In parallel to the development of various attention schemes, the surge of graph neural networks (GNNs) ignites a synchronous burst of studies employing GNNs of various architectures over spatial and temporal spaces, respectively or jointly, to handle spatial-temporal forecasting. GNNs feature modeling the local relationships, including proximity in space and time, while distinguishing the positions of the nodes (observations) with its asymmetric processing of the hub node and (temporal/spatial) neighboring nodes of different orders. Thereby, GNNs can naturally capture, at least

a portion of, the temporal and spatial information without integrating positional encodings. However, GNNs struggle with long-range dependencies due to the well-known issues of over-smoothing and over-squashing. The explanation for over-smoothing is that node representations become indistinguishable when the number of convolutional layers increases [54]. Over-squashing consists of the distortion of an exponential amount of information from distant nodes trying to pass through some edges, called "bottlenecks", in the graph [1].

We conjecture that this is because, unlike natural language, which inherently presents distinguishable structures in the data, such as being able to spot differences between any pair of sentences merely by reviewing their constituting words and the order among the words, time series data can have indistinguishable structures across many contexts. For instance, subsequences collected on Monday morning and Friday morning may have similar upward trends but are associated with different contexts in terms of the day of the week. If the model struggles to discern the divergence between subsequences, it will encounter severe difficulty in modeling the structure of time series data, which involves not only putting together subsequences of the same semantic but also putting apart subsequences of different semantics. Moreover, the strong capacity of deep learning models can exacerbate this problem, increasing the risk of overfitting, as the model strives to probe every subtle and spurious difference that results from variations in noise rather than patterns. We conduct a theoretical investigation into this problem in our studies, so we leave further discussion for the main part of our thesis.

To address the above issue, the core idea is to leverage domain knowledge, which can explicitly recover the structure of time series data, strengthening the

correlation between subsequences corresponding to the same context and attenuating correlations between samples corresponding to different contexts.

2.4.1 Spatially-Enhanced GNN

This subsection delves into the nuances of enhancing Graph Neural Networks (GNNs) with spatial correlations. Spatial correlations in data can manifest in various forms, and understanding these forms is crucial for effective modeling. The first form is geographical closeness, which is well-captured by Tobler's first law of geography: "Everything is related to everything else, but near things are more related than distant things" [84]. Techniques like Convolutional Neural Networks (CNNs) [118, 117, 111], Graph Convolutional Networks (GCNs) [113, 56], and Graph Attention Networks (GATs) [37, 31, 127, 105] have been particularly effective in modeling this type of spatial correlation. The second form is connectiveness, where regions connected by main roads or with high volumes of transitions between them are strongly correlated [28, 7, 95, 111]. The third form is functional similarity, where regions with similar functions, such as business districts, exhibit similar traffic patterns. This has been modeled using local point-of-interest (POI) statistics and GCNs [60, 28]. Recent advancements in this area include the application of meta-learning techniques to develop custom models for regions based on their specific functionalities [71, 72].

2.4.2 Temporally-Enhanced GNN

Temporal correlations are another critical aspect that can be integrated into GNNs for more robust and accurate models. Two primary types of temporal correlations are considered: recentness and periodicity. Recentness is based on the assumption that recent observations are more closely related to each other. To capture this,

various sequence modeling techniques have been employed, including ARIMA [70], LSTM [111, 110, 42, 109], attention networks [31, 105, 131], and Wavenet [104, 25]. Periodicity, on the other hand, refers to the recurring patterns in data, such as daily or weekly traffic patterns. Basic approaches to capture this include the engineering of temporal features [117, 85, 57]. More advanced methods incorporate this property directly into the hidden layers of deep neural networks [134, 10]. A growing body of work is focusing on the simultaneous modeling of both spatial and temporal correlations to capture more complex spatiotemporal patterns [9, 32].

2.4.3 Attention Enhanced GNN

We separate the literature on the attention mechanism from that of the Transformer, since studies integrating the attention mechanism into GNNs have a history that predates those employing Transformers. The attention mechanism has been widely applied to spatial-temporal data forecasting, especially in short-term forecasting, since 2018 [58, 31]. This trend started almost three years before the introduction of the Informer [129], which is primarily tested on long-term forecasting. In these earlier studies, researchers utilized spatial and temporal attention in various ways to capture spatial and temporal correlations. However, the complex and noisy nature of time series data can sometimes result in inaccurate attention scores, which could hinder the optimal utilization of historical spatial-temporal patterns crucial for predicting upcoming observations. Addressing this problem, GMAN [127] is, to the best of our knowledge, the pioneering work that integrates spatial and temporal embeddings with time series data to enhance the impact of spatial and temporal attention. These embeddings serve as an inductive bias, introducing prior information for attention weights between each pair of spatial-temporal points.

These priors are determined by the characteristics of the involved series as well as the properties of the associated time steps - for example, the time of day and day of the week. [14, 120, 34, 128] follows this line of research.

Chapter 3

Deciphering Dynamic Interdependencies in Urban Travel Demand

Forecasting urban travel demand is crucial for service providers, ranging from bus companies to ride-hailing platforms. While many studies have focused on individual transportation modes, our approach is more comprehensive, encompassing multiple transportation modes. We aim to integrate data from various transportation channels, delving into their co-evolving dynamics. These dynamics, deeply rooted in space-time-varying interdependencies, manifest as synchronized patterns that different modes display within specific temporal and spatial contexts.

Such co-evolution reflects the intricate space-time-varying interdependencies among transportation modes. The simultaneous evolution of these modes underscores their mutual interconnectedness, which varies both temporally and spatially. For instance, in a business district, the interdependencies might lead to a synchronized surge in demand across all transportation forms during evening rush hours, only to diminish post-rush hour. In contrast, entertainment districts, influenced

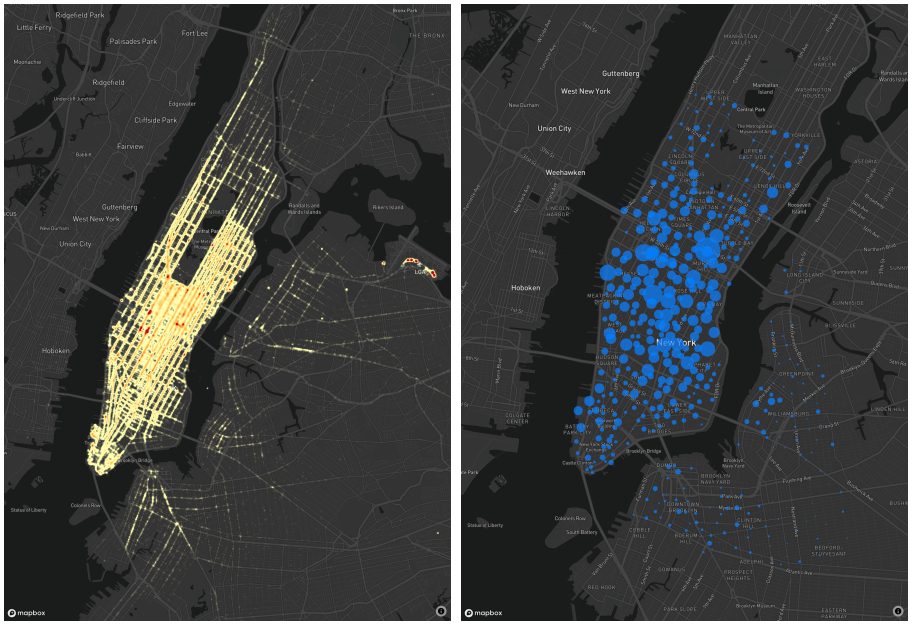
by different space-time factors, might experience an opposite trend, with demand peaking post-rush hours, mirroring the area's nighttime activities.

Our research introduces a groundbreaking co-evolving pattern learning framework, specifically tailored to decipher these space-time-varying interdependencies. This model excels at identifying dynamic correlations across the transportation spectrum and is also adept at capturing the nuanced interplay of demand influenced by both time and location. This highlights the significance of space-time-varying interdependencies in urban mobility forecasting.

3.1 Introduction

Urban transportation demand prediction is a real-world problem of growing importance. Online ride-sharing platforms like Didi, Uber, and Lyft heavily rely on demand prediction services to dynamically adjust their driver-customer matching strategies [106, 53]. Moreover, public transport operators, such as government transit agencies and private transport companies, can offer better service by allocating adequate transport capacities to different regions, aiming to match supply with demand [29]. Thus, predicting future transportation demand volume is desirable. Typically, past transportation volume data is used for this purpose. To accurately predict demand in the near future, it is crucial to focus on short-term patterns [111, 118, 117]. However, identifying and extracting these patterns can be challenging due to the potential fluctuations in short-term demand. A temporary deviation might cause the model to misinterpret a pattern change, leading to inaccurate predictions. In this paper, we discuss forecasting the future demand of one mode of transport (e.g., taxi) using demand data from another source (e.g., shared-bike). Here, **target demand** refers to the predicted demand, and **reference demand**

refers to the demand assisting the prediction. Notably, our model can incorporate reference demand from multiple sources, such as shared-bike, bus, and subway. However, due to data availability limitations, we focus on the relationship between taxi and shared-bike demand.



(a) NYC Yellow Taxi (b) NYC Citi Bike

Figure. 3.1 The distribution of taxi and shared-bike orders over New York City.

To address this, we posit that, for a specific district in a city during a certain time period, different types of transportation tend to exhibit similar trends. Leveraging behavior patterns from other transportation types can help eliminate random disturbances that might hinder pattern extraction. Once data from different transportation systems is collected, we can enhance the demand forecasting system for each mode.

To elucidate the relationship between taxi and bike demand, we visualize real data from New York City (NYC). Fig. 3.1 displays the starting points of taxi and

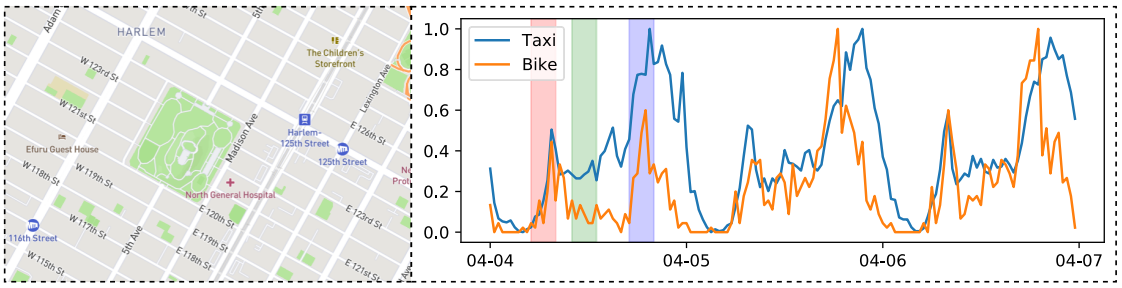


Figure. 3.2 An example region R. The left figure shows a park and a hospital in R, indicating residential use. The right figure shows the bike and taxi demand in R, exhibiting varying correlations throughout the day (colored in red, blue, and green).

bike trips over the city during a workday, while Fig. 3.2 shows the evolution of aggregate taxi and bike demand within an example region R over 3 days. From the spatial perspective, areas with high taxi demand often also have high bike demand. Temporally, during certain daily periods like rush hours, bike and taxi demand patterns often align. However, sometimes these patterns experience temporal shifts, as seen in the blue-highlighted time period.

Both spatial and temporal views suggest that taxi and bike demands are well-correlated in terms of scale and variation. Thus, they can act as a **pulse detector** for each other, providing insights into shared **co-evolving patterns**.

The challenge lies in determining the **co-evolving correlation**, a measure of the correlation between pairs of short-term patterns. A high value indicates that the patterns are aligned or co-evolving. However, deriving this correlation is challenging due to:

Diversity of co-evolving correlation. This correlation depends on the intrinsic attributes of the region and the time period. (1) Regional attributes reflect the active population in the region, influencing the correlation. For example, in an

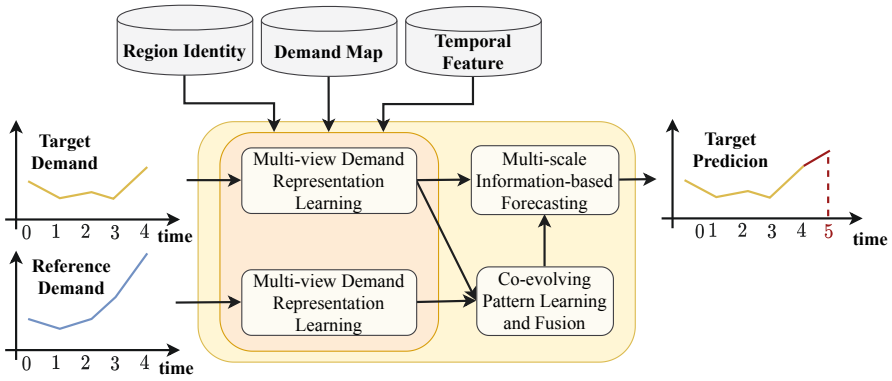


Figure. 3.3 Overview of the framework.

educational area, students might prefer cheaper transportation like shared bikes and buses over taxis. Conversely, business districts with office workers might see usage of both taxis and bikes. (2) Time period: Transportation preferences vary throughout the day, leading to different correlations. As seen in Fig. 3.2, bike and taxi demand correlate highly during morning and evening rush hours but less so during other times.

Multi-modal data fusion. Effective co-evolving correlation discovery involves multi-modal data, such as bike and taxi demand data, geographical data, and temporal data. These data types have different representations. For instance, bike and taxi demand data are numerical and evolve over time, while geographical data, like road networks, is static. Fusing these data types requires a unified framework, and different fusion methods can yield different forecasting results. A naive solution is to concatenate all features into a wide vector, but this overlooks temporal relationships among features and doesn't handle heterogeneity well. Thus, understanding the relationships between modalities is crucial for selecting an efficient fusion method.

To address these challenges, we introduce the co-evolving spatio-temporal neural network (CEST), outlined in Fig. 3.3. CEST first learns a multi-view representation for each demand type, combining periodic and real-time views. The periodic view captures the demand's recurring patterns, while the real-time view captures its irregularities. We obtain this representation hierarchically, first integrating periodic features (e.g., time of day) with region identities and real-time demand maps to derive periodic and real-time view representations. Next, we fuse these views. Co-evolving patterns are then extracted by matching target and reference demands across timestamps. These patterns are fused with the target demand's original patterns to create a calibrated pattern. Finally, we use a multi-scale representation, combining the calibrated pattern representation and raw demand value, to predict the next timestamp.

Several existing studies [41, 58, 112] address similar topics. However, our work distinguishes itself by encoding periodic features into demand representation and addressing temporal shifts commonly observed in real-world transportation systems.

Our contributions are:

- A multi-view demand representation learning module for each demand type, unifying periodic and real-time views.
- A co-evolving pattern learning and fusion module that extracts and fuses patterns to obtain a calibrated target demand pattern.
- Multi-scale information-based forecasting, combining original demand volume (fine-scale information) with calibrated pattern representation (coarse-scale information).

3.2 Preliminaries

In this section, we introduce the definitions and the problem statement. Frequently used notations are summarized in Table 3.1.

Definition 1. ***City Graph.*** We represent the city as an unweighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. Here, $\mathcal{V}$ denotes a set of predefined nodes representing grids or irregular regions. $\mathcal{E}$ is a set of edges indicating the connection between nodes. In our work, connections are based on geographical proximity, but this can be extended to other types of spatial relationships.

Definition 2. ***Demand.*** Demand at a node encompasses both the volume of inflow to that node and the volume of outflow from it. Assuming we have demand data from a target source and N_r reference sources, and there are N_l nodes in the city, each reporting d_{tg} types of information about the target demand and d_{rf} types of information about the reference demand over historical T timestamps. The target demand data is then represented as a tensor $\mathbf{X} \in \mathbb{R}^{T \times N_l \times d_{\text{tg}}}$. Similarly, the reference demand data is represented as $\hat{\mathbf{X}} = \{\hat{\mathbf{X}}^{(0)}, \cdots, \hat{\mathbf{X}}^{(N_r-1)}\}$. In our scenario, both d_{tg} and d_{rf} are set to 2, representing inflow and outflow data. The expected values for citywide target demand at the upcoming timestamp $T+1$ are denoted as $\mathbf{Y}_{T+1} \in \mathbb{R}^{N_l \times d_{\text{tg}}}$.

Definition 3. ***Co-evolving Correlation.*** Co-evolving correlation measures the relationship between the short-term patterns of target and reference demands.

Problem 1. Given the previous τ steps of demand data from the target source $\mathbf{X}_{T-\tau+1:T}$ and from the reference sources $\{\hat{\mathbf{X}}^{(0)}_{T-\tau+1:T}, \cdots, \hat{\mathbf{X}}^{(N_r-1)}_{T-\tau+1:T}\}$, the goal is to predict the expected values $\mathbf{Y}_{T+1}$ for the target demand at the next timestamp.

Table 3.1 Notations

Notation	Description
N_r, N_l, T	Number of reference sources / locations / timestamps.
$\mathbf{X} \in \mathbb{R}^{T \times N_l \times d_{\text{tg}}}$	Demand data collected from the target source.
$\hat{\mathbf{X}}^{(r)} \in \mathbb{R}^{T \times N_l \times d_{\text{rf}}}$	Demand data collected from the reference source r.
$\mathbf{Y}_{T+1} \in \mathbb{R}^{N_l \times d_{\text{tg}}}$	Actual demand at timestamp $T+1$.
$\bar{\mathbf{Y}}_{T+1} \in \mathbb{R}^{N_l \times d_{\text{tg}}}$	Predicted demand at timestamp $T+1$.
$\mathbf{E} \in \mathbb{R}^{T \times N_l \times d_h}$	Periodic-view representation.
$\mathbf{F} \in \mathbb{R}^{T \times N_l \times d_h}$	Real-time-view representation.
$\mathbf{H} \in \mathbb{R}^{T \times N_l \times d_h}$	Multi-view representation.
$\mathbf{S} \in \mathbb{R}^{T \times N_l \times d_h}$	Pattern representation.
$\mathbf{Z} \in \mathbb{R}^{T \times N_l \times d_h}$	Calibrated pattern representation.
$\mathbf{x}, \mathbf{y}, \bar{\mathbf{y}}, \mathbf{e}, \mathbf{f}, \mathbf{h}, \mathbf{s}, \mathbf{z}$	2d matrices that represent certain region.
$\hat{\mathbf{E}}^{(r)}, \hat{\mathbf{F}}^{(r)}, \hat{\mathbf{H}}^{(r)}, \hat{\mathbf{S}}^{(r)}$	Representations corresponding to the reference source r.

3.3 Co-evolving Spatio-temporal Neural Network

In this section, we detail each component of CEST, the architecture of which is depicted in Fig. 3.4.

3.3.1 Multi-view Demand Representation Learning

The multi-view demand representation considers both the periodic view and the real-time view. The periodic view pertains to the intrinsic attributes of a region, such as entertainment, residence, and business, which typically change periodically. Specifically, most regions exhibit different functionalities at various times of the day, leading to distinct human mobility patterns that influence transportation demand. For example, a Central Business District (CBD) functions as an office area during working hours but also offers shopping and dining services post-work. Furthermore, the dominant function of a region often follows a periodic pattern, where the same time across different days displays consistent dominant functions. While the periodic view can account for a significant portion of transportation

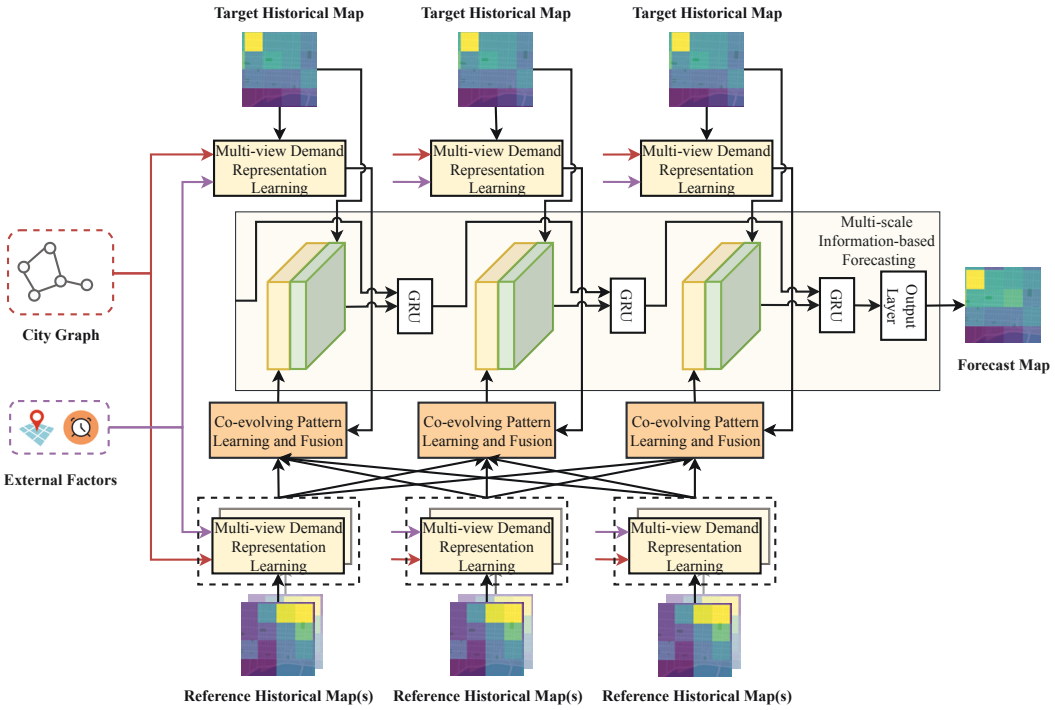


Figure. 3.4 The architecture of CEST. The top part illustrates the processing of target demand, the bottom part showcases the processing of reference demand from multiple sources, and the middle part highlights the multi-scale information-based temporal forecasting via GRU.

demand, there are irregular factors, like current weather conditions or events, that can't be captured without extensive external information. To address this, we introduce the real-time view. This view pertains to the real-time demand over a local area, encompassing the central region and its neighboring regions. This can somewhat reflect the aforementioned external factors. We separately learn the periodic-view and real-time-view demand representations and then fuse them to obtain a multi-view demand representation using our proposed cross-view self-attention block, which is an adaptation of the classical self-attention mechanism [88]. This workflow is illustrated in Fig. 3.5.

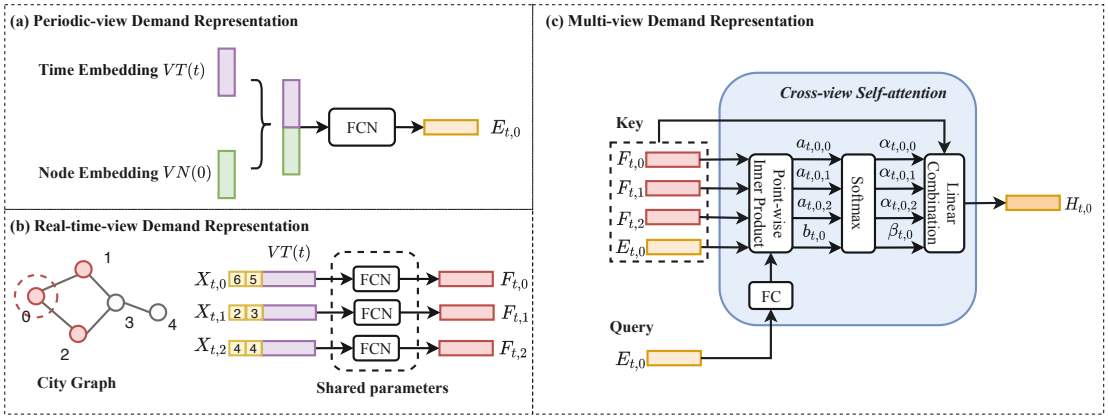


Figure. 3.5 An illustration of the workflow for learning multi-view demand representation over node 0 at time t. (a) Generation of the periodic-view demand representation through node and time embeddings. (b) Generation of the real-time-view demand representation using time embedding and demand value. (c) Fusion of the periodic-view and real-time-view demand representations to produce the multi-view demand representation.

The periodic-view demand representation is based on the current identity of the node, which can be interpreted as the current functionality of the corresponding region. Pan et al. [71] used POI data to reflect this identity. However, since POI data is not available in many cities, we draw inspiration from word2vec [66]. We create a lookup table VN for nodes and another lookup table VT for features related to periodicity. Specifically, VN stores node encodings:

$$VN = \{VN(1), \cdots, VN(N_l)\}, \tag{3.1}$$

where $VN(i) \in \mathbb{R}^{d_n}$ represents the encoding of node i. Regarding VT, it is further divided into sub-tables based on different periodicities, such as hour of the day and day of the week. Each sub-table stores encodings of different time displacements for

the corresponding periodicity. In our case, we consider daily and weekly periodicities, so VT comprises two sub-tables: VD (hour of the day) and VW (day of the week). The composition of VT is:

$$VT = \{VD, VW\}, \tag{3.2}$$

$$VD = \{VD(0), VD(1), \cdots, VD(23)\}, \tag{3.3}$$

$$VW = \{VW(0), VW(1), \cdots, VW(6)\}, \tag{3.4}$$

where both elements of the two sub-tables, $VD(h)$ and $VW(d)$, belong to $\mathbb{R}^{d_h}$.

VT operates as follows: upon receiving a real-time timestamp t, it first identifies the hour of the day and the day of the week associated with t. It then retrieves the encodings corresponding to the identified hour and day from their respective sub-tables. Finally, it aggregates the encodings using summation to obtain the encoding for timestamp t. This process can be formally described as:

$$t \rightarrow hour,\ day, \tag{3.5}$$

$$VT(t) = VD(hour) + VW(day). \tag{3.6}$$

Notably, all encodings in VT and VN are randomly initialized and are integrated into the model as free parameters, updated throughout the training process.

To derive the periodic-view demand representation of a node, we combine the node encoding with the periodic-feature encoding. Formally, for the periodic-view demand representation of node i at time t, denoted as $\mathbf{E}_{t,i} \in \mathbb{R}^{d_h}$, we retrieve the associated encoding of node i from VN and the associated encoding of time t from VT. We then combine them using a fully connected layer, as depicted in Fig. 3.5

(a). The formula is:

$$\mathbf{E}_{t,i} = \tanh\left(\mathbf{W}_{vn} VN(i) + \mathbf{W}_{vt} VT(t)\right), \tag{3.7}$$

where $\mathbf{W}_{vn} \in \mathbb{R}^{d_h \times d_n}$ and $\mathbf{W}_{vt} \in \mathbb{R}^{d_h \times d_t}$ are parameters of the fully connected layer.

The real-time-view demand representation of a node is derived based on the real-time demand over the node itself and its neighboring nodes in the city graph, reflecting spatial locality. We also integrate features related to periodicity with the real-time demand to provide contextualized semantics. As shown in Fig. 3.5 (b), the embedding for individual node i at timestamp t, denoted by $\mathbf{F}_{t,i} \in \mathbb{R}^{d_h}$, is:

$$\mathbf{F}_{t,i} = \tanh\left(\mathbf{W}_x \mathbf{X}_{t,i} + \mathbf{W}'_{vt} VT(t)\right), \tag{3.8}$$

where $\mathbf{W}_x \in \mathbb{R}^{d_h \times d_{\text{tg}}}$ (recall that d_{tg} is the dimension of the input feature for target demand) and $\mathbf{W}'_{vt} \in \mathbb{R}^{d_h \times d_t}$ are parameters. The real-time-view representation is then composed of the embeddings for individual nodes in $\mathcal{N}_i$, where $\mathcal{N}_i$ includes the neighborhoods of i and i itself. This is formally expressed as $\mathbf{SF}_{t,i} = \{\mathbf{F}_{t,j} \mid j \in \mathcal{N}_i\}$.

Next, we fuse the periodic-view representation with the real-time-view representation through the cross-view self-attention block, as shown in Fig. 3.5 (c). The periodic-view captures the regular pattern that presents itself repeatedly on a daily or weekly cycle, while the real-time-view captures the real-time pattern. Although the real-time-view contains comprehensive information regarding the demand at that timestamp, some of it is random disturbance that doesn't impact future demand. The periodic-view is also not perfect, as it lacks information about the current state. We filter out the useful information from both views in parallel and then fuse them. We achieve this by proposing a cross-view self-attention block. The query is the periodic-view representation $\mathbf{E}_{t,i}$, and the keys consist of the elements

in the real-time-view representation $\mathbf{SF}_{t,i}$ as well as the periodic-view representation $\mathbf{E}_{t,i}$ itself. This allows the model to automatically learn the importance weights of the periodic and real-time views. Let $\alpha_{t,i,j}$ be the attention weight for the element related to node j in $\mathcal{N}_i$, and $\beta_{t,i}$ be the attention weight for the periodic-view representation of node i itself. We compute them as:

$$\alpha_{t,i,j} = \frac{\exp(a_{t,i,j})}{\sum_{j \in \mathcal{N}_i} \exp(a_{t,i,j}) + \exp(b_{t,i})} \; \forall j \in \mathcal{N}_i, \tag{3.9}$$

$$\beta_{t,i} = \frac{\exp(b_{t,i})}{\sum_{j \in \mathcal{N}_i} \exp(a_{t,i,j}) + \exp(b_{t,i})}, \tag{3.10}$$

where

$$a_{t,i,j} = \mathbf{E}_{t,i}{}^{\top} \mathbf{U}_d \mathbf{F}_{t,j}, \; \forall j \in \mathcal{N}_i, \tag{3.11}$$

$$b_{t,i} = \mathbf{E}_{t,i}{}^{\top} \mathbf{U}_d \mathbf{E}_{t,i}. \tag{3.12}$$

Here, $\mathbf{U}_d \in \mathbb{R}^{d_h \times d_h}$ is a trainable weight matrix. The multi-view representation $\mathbf{H}_{t,i}$ is then calculated by the linear combination of the periodic-view and real-time-view representations, weighted by the attention scores:

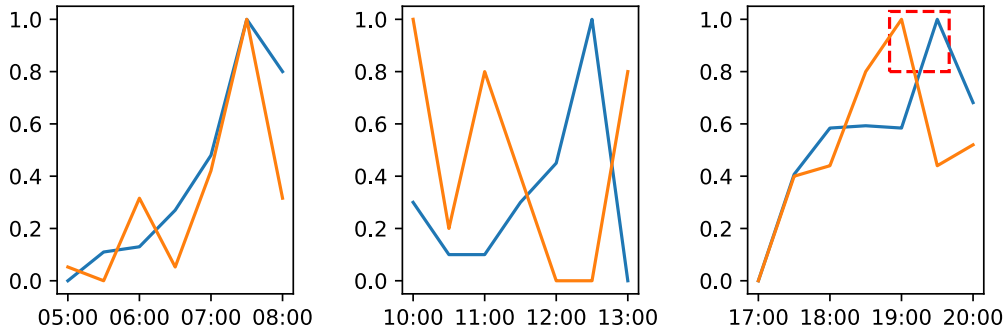
$$\mathbf{H}_{t,i} = \sum_{j \in \mathcal{N}_i} \alpha_{t,i,j} \mathbf{F}_{t,j} + \beta_{t,i} \mathbf{E}_{t,i}, \tag{3.13}$$

In the cross-view self-attention block, we treat the periodic-view and real-time-view representations equally, as the learning process for these two representations adequately projects both into the same latent space.

The representation of the demand from reference source r is computed similarly and is denoted as $\hat{\mathbf{H}}^{(r)}_{t,i}$. Notably, we create separate lookup tables $\hat{VT}^{(r)}$ and $\hat{VN}^{(r)}$ for each source r, since the demand for different transportations exhibits distinct evolving patterns throughout the day. For example, the shared-bike demand

between 10:00 and 17:00 is relatively lower than the taxi demand, as shown in Fig. 3.2.

3.3.2 Co-evolving Pattern Learning and Fusion



(a) 5:00-8:00 (b) 10:00-13:00 (c) 17:00-20:00

Figure. 3.6 Detailed comparison of bike demand and taxi demand in region R over different time ranges, following Fig. 3.2. Taxi demand and bike demand are separately normalized between $[0, 1]$ in different time windows for comparison.

A co-evolving pattern represents the short-term trends observed concurrently in the evolution of both target and reference demands. Two challenges arise when extracting this co-evolving pattern: 1) A short-term pattern is constructed from multiple consecutive observations. 2) The degree of co-evolution between taxi and bike demands varies throughout the day, sometimes exhibiting temporal drifts, as depicted in Fig. 3.6.

To address the first challenge, we generate a set of short-term pattern representations for each demand, summarizing different evolutionary segments. For the second challenge, we utilize the cross-view self-attention block, as introduced in Sec. 3.3.1. This block computes co-evolving correlations between key-query matching

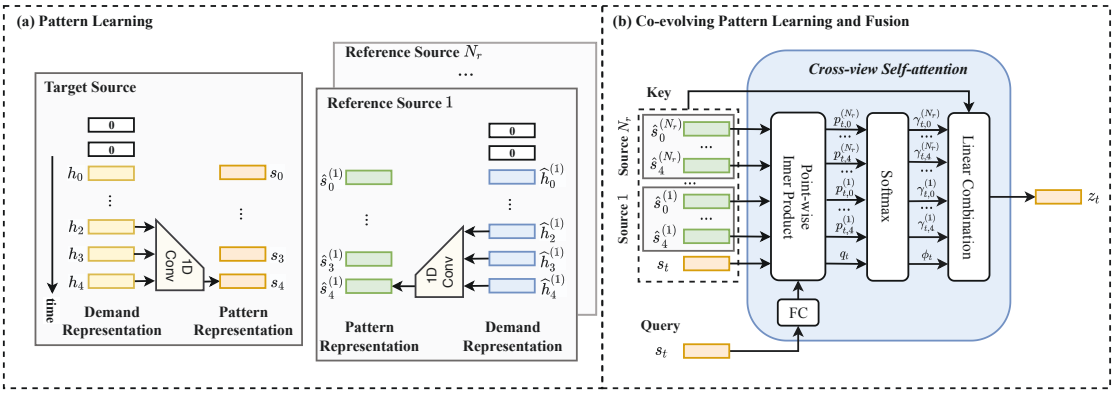


Figure. 3.7 (a) An illustration of pattern learning. (b) A network design for learning co-evolving patterns and fusing them with the target demand patterns at each timestamp.

patterns, extracts co-evolving patterns, and then fuses the co-evolving patterns with target patterns in a weighted manner to produce calibrated target patterns.

For short-term pattern extraction, we group temporally adjacent frames to form a segment and encode them into a unified representation. Specifically, we apply a 1D convolution along the time axis followed by an activation function to derive the pattern representation. We choose a kernel size of 3 for both target and reference demands and apply zero padding of size 2 to the sequence's left tail to maintain consistency between input and output lengths. This process is illustrated in Fig. 3.7 (a). For simplicity, we use lowercase notation to denote the representation of a specific region. Let $\mathbf{s}_t$ represent the pattern of a certain region from $t-2$ to t. It is defined as:

$$\mathbf{s}_t = \tanh\left(\mathbf{W}_s^1 \mathbf{h}_{t-2} + \mathbf{W}_s^2 \mathbf{h}_{t-1} + \mathbf{W}_s^3 \mathbf{h}_t\right), \tag{3.14}$$

where $\mathbf{W}_s^1 \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{W}_s^2 \in \mathbb{R}^{d_h \times d_h}$, and $\mathbf{W}_s^3 \in \mathbb{R}^{d_h \times d_h}$ are parameters. $\mathbf{h}_t$ is the multi-view demand representation of this region, as derived in Sec. 3.3.1. The

pattern representation for each reference demand is similarly computed using a distinct parameter set, denoted as $\hat{\mathbf{s}}_t^{(r)}$.

Next, we jointly conduct co-evolving pattern learning and fusion, as depicted in Fig. 3.7 (b). By leveraging the cross-view self-attention block introduced earlier, the model can determine the optimal combination of reference patterns to compose the co-evolving pattern. It can also find the best combination of the co-evolving pattern and the target pattern itself. In our approach, we use the entire set of patterns from different reference sources as keys, irrespective of the timespan, to address the issue of temporal drift, given that we only consider a short span of historical data. Specifically, we denote the attention weight over the pattern from reference source r at timestamp t' as $\gamma_{t,t'}^{(r)}$, and the attention weight over the target pattern itself as ϕ_t. Let $\mathcal{N}_t = \{t-\tau+1, \cdots, t\}$ be the set of considered timestamps, and $\mathcal{R} = \{1, \cdots, N_r\}$ be the set of indices for different reference sources. The computation of $\gamma_{t,t'}^{(r)}$ and ϕ_t is as follows:

$$\gamma_{t,t'}^{(r)} = \frac{\exp(p_{t,t'}^{(r)})}{\sum_{r\in\mathcal{R}}\sum_{t'\in\mathcal{N}_t}\exp(p_{t,t'}^{(r)}) + \exp(q_t)},\ \forall t' \in \mathcal{N}_t,\ r \in \mathcal{R}, \tag{3.15}$$

$$\phi_t = \frac{\exp(q_t)}{\sum_{r\in\mathcal{R}}\sum_{t'\in\mathcal{N}_t}\exp(p_{t,t'}^{(r)}) + \exp(q_t)} \tag{3.16}$$

where

$$p_{t,t'}^{(r)} = \mathbf{s}_t{}^\top \mathbf{U}_p \hat{\mathbf{s}}_{t'}^{(r)},\ \forall t' \in \mathcal{N}_t,\ r \in \mathcal{R}, \tag{3.17}$$

$$q_t = \mathbf{s}_t{}^\top \mathbf{U}_p \mathbf{s}_t, \tag{3.18}$$

with $\mathbf{U}_p \in \mathbb{R}^{d_h \times d_h}$ being a trainable weight matrix.

Finally, the representation of the calibrated pattern, $\mathbf{z}_t \in \mathbb{R}^{d_h}$, is computed as the average of the pattern representations from the target source and the reference

sources, weighted by their respective attention scores:

$$\mathbf{z}_t = \sum_{r \in \mathcal{R}} \sum_{t' \in \mathcal{N}_t} \gamma_{t,t',i} \hat{\mathbf{s}}_{t'}^{(r)} + \phi_t \mathbf{s}_t. \tag{3.19}$$

3.3.3 Multi-scale Information-based Forecasting

We concatenate the calibrated pattern representation $\mathbf{z}_t$ with the real-time demand volume $\mathbf{x}_t$ and feed the combined representation into a gated recurrent unit (GRU) [11]. The coarse-scale and fine-scale representations can mutually regulate the information flow to the long-term memory. For simplicity, we omit the node index. Let $\mathbf{m}_t \in \mathbb{R}^{d_g}$ represent the memory state at time t. The unit takes $\mathbf{z}_t$, $\mathbf{x}_t$, and the previous state $\mathbf{m}_{t-1}$ as inputs and produces the current state $\mathbf{m}_t$. The reset gate $\mathbf{r}_t \in \mathbb{R}^{d_g}$ determines the influence of $\mathbf{m}_{t-1}$ on the intermediate state $\mathbf{c}_t$. The update gate $\mathbf{u}_t \in \mathbb{R}^{d_g}$ controls the contribution of the intermediate state $\mathbf{c}_t$ to $\mathbf{m}_t$. The current state can be calculated as:

$$\mathbf{g}_t = \left[\mathbf{z}_t^\top, \mathbf{x}_t^\top\right]^\top \tag{3.20}$$

$$\mathbf{u}_t = \sigma\left(\mathbf{W}_g^u \mathbf{g}_t + \mathbf{W}_m^u \mathbf{m}_{t-1}\right), \tag{3.21}$$

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_g^r \mathbf{g}_t + \mathbf{W}_m^r \mathbf{m}_{t-1}\right), \tag{3.22}$$

$$\mathbf{c}_t = \tanh\left(\mathbf{r}_t \circ \mathbf{m}_{t-1} + \mathbf{W}_g^c \mathbf{g}_t\right), \tag{3.23}$$

$$\mathbf{m}_t = \mathbf{u}_t \circ \mathbf{m}_{t-1} + (1 - \mathbf{u}_t) \circ \mathbf{c}_t, \tag{3.24}$$

where $\circ$ denotes the Hadamard product, and $\sigma(\cdot)$ is the sigmoid function. The matrices $\mathbf{W}_g^u$, $\mathbf{W}_m^u$, $\mathbf{W}_g^r$, $\mathbf{W}_m^r$, and $\mathbf{W}_g^c$ are the GRU weights.

The predicted demand value for the subsequent timestamp is given by:

$$\bar{\mathbf{y}}_{t+1} = \mathbf{W}_o \mathbf{m}_t, \tag{3.25}$$

where $\mathbf{W}_o \in \mathbb{R}^{d_{\text{tg}} \times d_g}$ is the output projection matrix. The predictions $\bar{\mathbf{y}}_{t+1}$ from different nodes are aggregated to form the citywide prediction $\bar{\mathbf{Y}}_{t+1}$. The training objective is to minimize the mean squared error between the predicted and actual values:

$$\mathcal{L} = \frac{1}{N_l} ||\bar{\mathbf{Y}}_{t+1} - \mathbf{Y}_{t+1}||^2. \tag{3.26}$$

We use the Adam optimizer [47] to minimize this objective.

3.4 Evaluation

3.4.1 Experimental Setting

Datasets

We validate our model's effectiveness by conducting experiments on two renowned cities: Chicago and New York City. The study area in NYC is bounded by the coordinates (40.68°, −73.99°), (40.77°, −73.93°), (40.79°, −73.98°), and (40.70°, −74.04°). In Chicago, the boundaries are (41.73°, −87.54°), (42.01°, −87.54°), (42.01°, −87.74°), and (41.73°, −87.74°). For each city, we gather taxi and shared bike trip data and design two tasks, where taxis and bikes alternately serve as the target and reference transportation modes. The study period spans from April 1, 2016, to June 30, 2016. Each dataset's details are as follows:

- **NYC Citi Bike**: The NYC Bike Sharing System, Citi Bike, provides order data on its official website[1]. During the specified period, approximately 3.7 million transaction records were generated. Each record contains information about pick-up and drop-off locations and times.

[1]https://www.citibikenyc.com/system-data

- **NYC Yellow Taxi**: The NYC Yellow Taxi dataset comprises around 35 million taxicab trip records in New York[2]. On average, about 380,000 trip records are generated daily. This dataset also includes information similar to the NYC Citi Bike dataset.

- **Chicago Bike**: The Divvy bike sharing system in Chicago provides trip data on its website[3]. An average of 11,768 trips were recorded daily during the study period.

- **Chicago Taxi**: The Chicago taxi data, provided by the local government on the Chicago Data Portal[4], indicates an average of 72,104 transaction records per day based on our analysis.

Task Description

We partition the study area of NYC into a 7×14 grid and construct a city graph based on the grid's geographical properties. Specifically, each cell in the grid corresponds to a node in the graph. The neighbors of a cell in the graph are defined as the cells that are vertically, horizontally, or diagonally adjacent to it in the grid. We apply zero padding to the grid's boundary. It's worth noting that our approach is also applicable in environments with irregular topologies, where the number of neighbors varies. The time interval is set to half an hour. We then determine the inflow and outflow volumes for bikes and taxis in each region. We normalize the demand values of taxis and bikes to the range $[-1, 1]$ to facilitate easier identification of their relationships. For Chicago, the settings are similar, but the grid size is 17×30 due to the larger study area compared to NYC.

[2]https://data.cityofnewyork.us/Transportation/2016-Yellow-Taxi-Trip-Data/k67s-dv2t
[3]https://www.divvybikes.com/system-data
[4]https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew

For prediction, we use the six most recent observations, covering three hours. We randomly partition the first eighty days into disjoint training and validation sets at a ratio of $7:1$, while the last ten days form the testing set.

Network Setting

We implement the network using the PyTorch toolkit. The network architecture tested in the experiment is depicted in Fig. 3.4. In the multi-view representation learning module, the dimensions of the periodic-feature embedding d_t, node embedding d_n, and demand representation d_h are all set to 8. In the co-evolving patterns learning module, the dimension of the short-term pattern representation d_h is also 8. Furthermore, we use a one-layer GRU with 8 hidden channels for multi-scale information-based forecasting. We also tested other hyper-parameter settings to demonstrate our model's robustness. The learning rate for the Adam optimizer is set to 0.0002, and other hyper-parameters use the default settings in PyTorch. Parameters in the network are initialized using PyTorch's default settings. We update the model parameters iteratively through batch training until convergence, with a batch size of 8.

3.4.2 Evaluation Metrics

We evaluate our algorithm using root mean square error (RMSE) and mean absolute error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|, \tag{3.27}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}, \tag{3.28}$$

where n is the total number of instances, y_i represents the ground truth value, and $\hat{y}_i$ is the predicted value.

3.4.3 Baselines

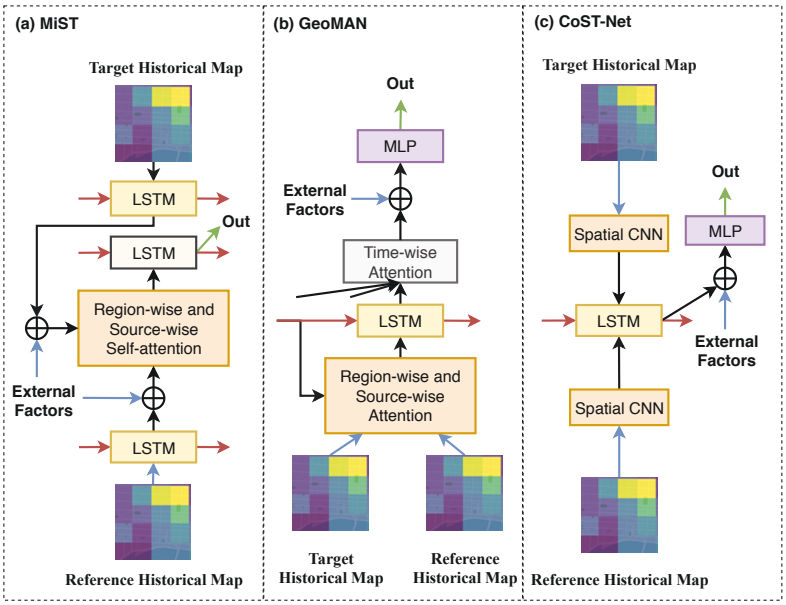


Figure. 3.8 The architectures of three baseline models: (a) MiST, (b) GeoMAN, and (c) CoST-Net. We provide a skeleton for each to illustrate its core idea. For more detailed implementation, we refer readers to their original papers [112, 58, 41]. In each figure, blue lines represent input flows; green lines represent output flows; and red lines represent recurrent flows. In the region-wise and source-wise attention modules of MiST and GeoMAN, representations of specific regions are derived by applying attention over neighboring regions and different sources. In GeoMAN's time-wise attention module, it focuses on representations at historical timestamps to make predictions for the current timestamp.

- **HA.** Historical Average predicts future demand using the average of historical values from the same time interval.

- **ARIMA.** Autoregressive Integrated Moving Average (ARIMA) is a linear model. We build and train an ARIMA model for each region using the ARIMA module from the statsmodel Python package, setting the orders to $(6, 1, 0)$.

- **GBRT**. Gradient Boosted Regression Trees (GBRT) is a non-parametric statistical learning technique for regression. The input feature comprises the historical demand of the target region and its neighboring regions. We use the GBRT implementation from the sklearn Python package. Key parameters for GBRT are set as follows: $n\_estimators = 500, max\_depth = 3, min\_samples\_leaf = 1$. All other parameters use default settings.

- **STResNet** [117]. A state-of-the-art model for spatio-temporal forecasting, STResNet models spatio-temporal correlations using residual units. Our input is a four-channel tensor, with each channel representing a type of demand: taxi inflow, taxi outflow, bike inflow, and bike outflow. The ResNet architecture consists of 2 residual blocks with 8 hidden units each. We concatenate the representation from each region output from ResNet's final layer with an 8-dimensional time embedding and map the concatenation to the prediction using a 1-layer fully-connected network.

- **MiST** [41]. MiST, depicted in Fig. 3.8 (a), is a multi-source spatial-temporal forecasting model with an attention mechanism. Its primary distinction from GeoMAN is its use of event encodings (or transportation encodings in our context) and region encodings when computing attention weights. We set the dimension of these encodings to 8. Additionally, two recurrent networks encode sequences at different levels, each implemented using an LSTM with 8 hidden units.

- **GeoMAN** [58]. As illustrated in Fig. 3.8 (b), GeoMAN is designed for multi-source spatial-temporal data, applying attention over different data sources, spatial neighbors, and timestamps. In our context, it assigns attention weights to taxi and bike demands at each timestep to identify the more crucial signal for forecasting. We use an LSTM with 8 hidden units for sequential modeling. Unlike the original work, we perform "global attention" on each region's spatial neighbors since long-range relationships are weak in our context.

- **CoST-Net** [112]. CoST-Net, shown in Fig. 3.8 (c), models multi-source demand by constructing a wide recurrent neural network. The input is the concatenation of representations of each demand type within a patch of regions. Specifically, we obtain each representation using a 1-layer deep convolutional encoder with 4 filters. A 1-layer LSTM with 16 hidden units (due to our 4 demand types) encodes the sequence. We also encode the temporal feature into the sequence's high-level representation similarly to STResNet. Finally, each demand type is decoded by a 1-layer deep convolutional network, which inversely processes the encoder to make the forecast.

For all baseline models, input features match those of our proposed model, including taxi inflow, taxi outflow, shared-bike inflow, shared-bike outflow, and external features like the hour of the day and day of the week. We also employ the Adam optimizer [47] with the same setup as our model.

3.4.4 Experiment Results

We present the detailed experimental results for NYC in Table 3.2, and for Chicago in Table 3.3. Specifically, regarding the repeated tests in NYC, we analyze detailed statistics, including the mean and standard deviation, from both temporal and

Table 3.2 Performance on NYC tasks.

Models			HA	ARIMA	GBRT	STResNet	GeoMAN	CoST-Net	MiST	CEST
Taxi	overall	MAE	13.74	9.60	7.45	7.82 ± 0.22	8.27 ± 0.11	8.02 ± 0.03	7.54 ± 0.03	$\mathbf{7.09 \pm 0.03}$
		RMSE	27.87	17.50	13.37	13.84 ± 0.37	14.72 ± 0.16	14.04 ± 0.06	13.51 ± 0.05	$\mathbf{12.62 \pm 0.04}$
	0:00∼6:00	MAE	14.28	7.55	4.68	5.49 ± 0.25	5.29 ± 0.07	5.19 ± 0.07	4.84 ± 0.03	$\mathbf{4.60 \pm 0.02}$
		RMSE	29.44	13.36	9.02	9.88 ± 0.31	9.90 ± 0.13	9.57 ± 0.08	9.17 ± 0.05	$\mathbf{8.63 \pm 0.05}$
	6:00∼12:00	MAE	13.92	9.66	7.57	7.87 ± 0.19	8.80 ± 0.13	8.66 ± 0.04	7.92 ± 0.05	$\mathbf{7.32 \pm 0.04}$
		RMSE	31.95	18.01	13.26	13.84 ± 0.41	14.93 ± 0.23	14.72 ± 0.21	13.84 ± 0.13	$\mathbf{12.81 \pm 0.14}$
	12:00∼18:00	MAE	11.82	9.30	7.77	7.94 ± 0.23	7.98 ± 0.09	7.82 ± 0.03	7.52 ± 0.04	$\mathbf{7.01 \pm 0.03}$
		RMSE	20.62	16.02	12.83	13.21 ± 0.45	13.12 ± 0.11	12.55 ± 0.07	12.51 ± 0.07	$\mathbf{11.56 \pm 0.07}$
	18:00∼24:00	MAE	14.92	11.88	9.75	10.00 ± 0.19	11.02 ± 0.14	10.43 ± 0.02	9.81 ± 0.05	$\mathbf{9.35 \pm 0.03}$
		RMSE	28.18	21.56	17.12	17.39 ± 0.33	19.34 ± 0.20	17.97 ± 0.06	17.26 ± 0.07	$\mathbf{16.30 \pm 0.06}$
Bike	overall	MAE	3.69	3.33	2.44	2.62 ± 0.038	2.59 ± 0.042	2.50 ± 0.015	2.44 ± 0.005	$\mathbf{2.29 \pm 0.006}$
		RMSE	7.57	6.20	4.29	4.59 ± 0.088	4.56 ± 0.057	4.37 ± 0.012	4.35 ± 0.013	$\mathbf{4.03 \pm 0.014}$
	0:00∼6:00	MAE	0.72	1.42	0.69	1.04 ± 0.023	0.73 ± 0.007	0.72 ± 0.011	0.68 ± 0.005	$\mathbf{0.66 \pm 0.003}$
		RMSE	1.48	2.18	1.29	1.57 ± 0.027	1.33 ± 0.014	1.31 ± 0.018	1.25 ± 0.012	$\mathbf{1.22 \pm 0.012}$
	6:00∼12:00	MAE	5.12	4.07	2.82	2.95 ± 0.052	3.13 ± 0.078	3.01 ± 0.027	2.87 ± 0.021	$\mathbf{2.62 \pm 0.013}$
		RMSE	9.51	7.45	4.65	4.98 ± 0.146	5.08 ± 0.091	4.88 ± 0.015	4.75 ± 0.020	$\mathbf{4.32 \pm 0.019}$
	12:00∼18:00	MAE	4.51	4.24	3.32	3.40 ± 0.050	3.34 ± 0.046	3.34 ± 0.012	3.34 ± 0.009	$\mathbf{3.09 \pm 0.005}$
		RMSE	8.31	7.46	5.27	5.49 ± 0.096	5.57 ± 0.058	5.27 ± 0.022	5.37 ± 0.010	$\mathbf{4.88 \pm 0.014}$
	18:00∼24:00	MAE	4.42	3.60	2.92	3.09 ± 0.054	3.00 ± 0.039	2.92 ± 0.014	2.88 ± 0.003	$\mathbf{2.79 \pm 0.006}$
		RMSE	8.23	6.14	4.77	5.19 ± 0.107	4.96 ± 0.051	4.82 ± 0.011	4.76 ± 0.028	$\mathbf{4.58 \pm 0.017}$

spatial perspectives. For Chicago, we only display the mean overall error for brevity. Notably, HA, ARIMA, and GBRT produce consistent forecasting results, so their standard deviation is 0. This is omitted in Table 3.2 for brevity. The reasons are: HA solely computes the average of historical records as predictions without involving any parameters; ARIMA assumes a linear relationship between past and future timesteps, resulting in a unique parameter solution; GBRT constructs a

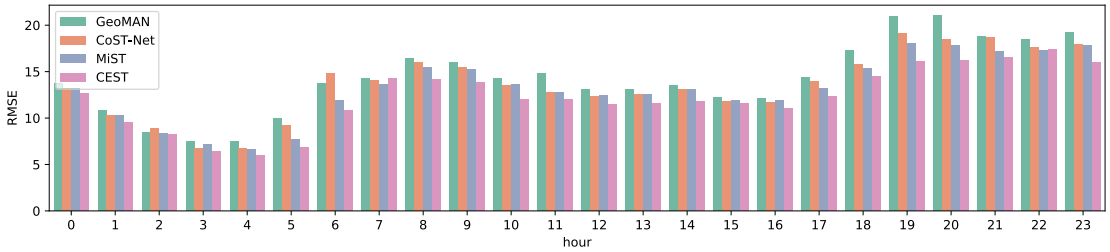
Table 3.3 Performance on Chicago tasks.

Models	Taxi		Bike	
	MAE	RMSE	MAE	RMSE
HA	0.99	6.49	0.39	1.91
ARIMA	0.62	3.05	0.36	1.55
GBRT	0.59	2.65	0.33	1.27
STResNet	0.68	2.65	0.36	1.27
GeoMAN	0.74	2.79	0.33	1.24
CoST-Net	0.64	2.45	0.34	1.16
MiST	0.59	2.55	0.32	1.22
CEST	**0.55**	**2.36**	**0.28**	**1.11**

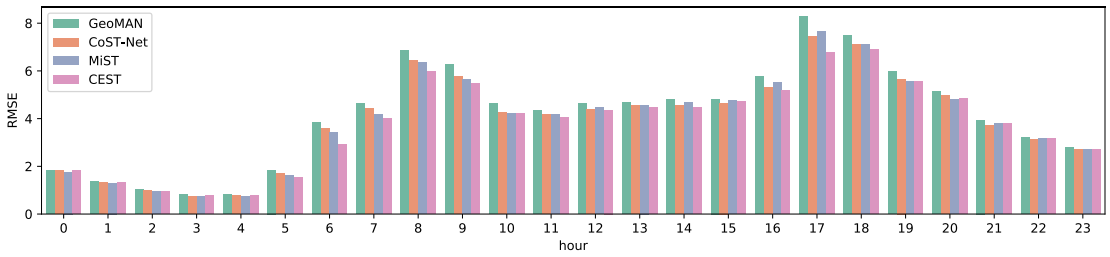
set of regression trees based on the entire training samples using a deterministic criterion, ensuring the model remains unchanged each time.

In the NYC taxi demand task, CEST surpasses the state-of-the-art model, MiST, by 6.0% and 6.6% in overall MAE and RMSE, respectively. For bike demand forecasting, CEST also demonstrates superior performance, outperforming the state-of-the-art model by 6.2% in MAE and 7.4% in RMSE. Traditional methods like HA and ARIMA underperform because they fail to model the complex non-linear spatial and temporal relationships. GBRT's performance heavily depends on the number of parameters. Compared to deep models, none of the previous approaches extract co-evolving patterns between taxi and bike demands. Moreover, except for MiST, other methods neglect region identity. For MiST, attention scores are computed based on demand correlation without considering pattern matching, making outcomes susceptible to temporal drift. GeoMAN considers the entire sequence when applying attention but treats taxi and bike demands equally, overlooking their heterogeneity. Both CoST-Net and STResNet don't utilize attention mechanisms, potentially incorporating irrelevant features into target demand.

Additionally, we compare model performance across different times of the day. We divide a day into four time periods: 0:00~6:00, 6:00~12:00, 12:00~18:00, and 18:00~24:00. Our model outperforms baseline models in all four periods, with the most significant improvements observed between 6:00~12:00 and 12:00~18:00. During 6:00~12:00, the MAE for taxi demand decreases by 7.3%, and the bike demand MAE drops by 8.8%. Between 12:00~18:00, the MAE for taxi demand reduces by 6.8%, and the bike demand MAE decreases by 7.6%. These improvements can be attributed to both bike and taxi demands experiencing peak emergence and decline during these periods, resulting in strong co-evolving correlations that enhance



(a) NYC Yellow Taxi.



(b) NYC Citi Bike.

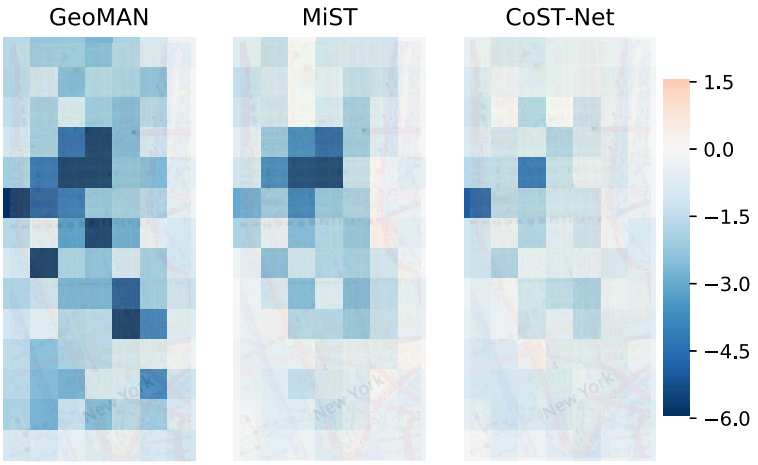
Figure. 3.9 Hourly comparison.

performance. For other periods, the correlation is less pronounced, especially for bike demand, due to lower rental frequencies at night. We also provide a detailed hourly comparison in Fig. 3.9.

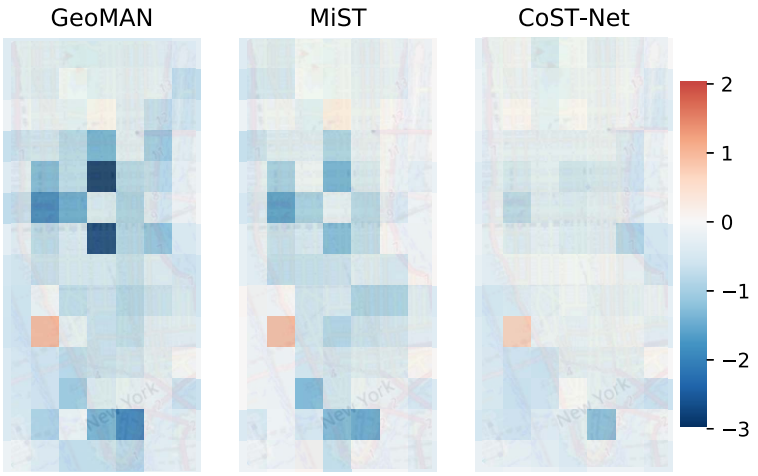
Furthermore, we assess our model's performance from a spatial perspective. The improvement over three baseline models at a regional level is depicted in Fig. 3.10. Our model consistently outperforms baseline models in nearly all regions.

3.4.5 Ablation Study

To understand the impact of each component in our model, we design four variants. Each substitutes one or more components of our model with commonly used modules. We evaluate their performance on NYC tasks and briefly describe these variants:



(a) NYC Yellow Taxi.



(b) NYC Citi Bike.

Figure. 3.10 Comparison from spatial view.

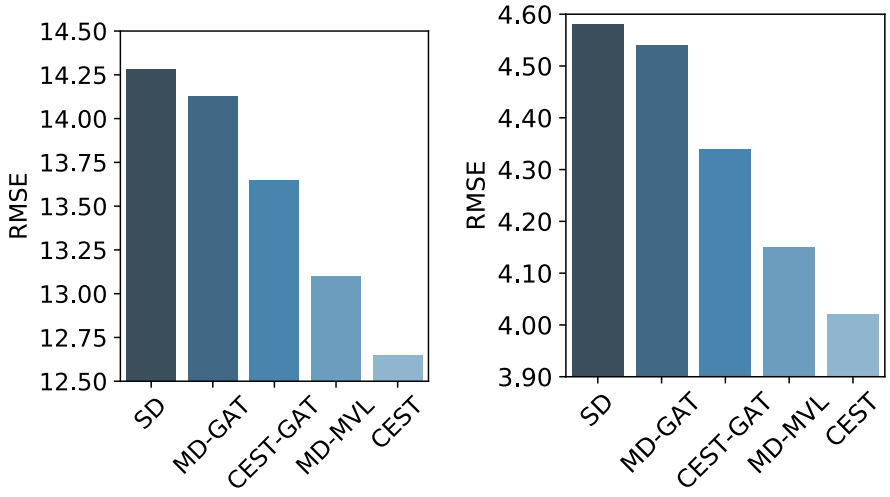
- SD: SD (single demand) only considers the target demand as input and uses GRU to model the temporal relationship. Temporal features are also encoded into the demand to obtain real-time demand representation. This variant aims to verify the significance of reference demand in forecasting.

- MD-GAT: MD-GAT (multi-demand) considers both target and reference demands. It employs GAT [89] for demand representation learning, an alternative to our multi-view demand representation learning (MVL) module. A GRU encodes the concatenation of target and reference demands in time order, but without extracting co-evolving patterns.

- CEST-GAT: CEST-GAT uses GAT for demand representation learning and subsequently extracts co-evolving patterns as described in Sec. 3.3.2.

- MD-MVL: MD-MVL obtains demand representation using the MVL module as per Sec. 3.3.1. It also encodes the sequence of demand representations using GRU.

The evaluation results are presented in Fig. 3.11. Each proposed component enhances performance, with a combination of these components delivering the best results. The MVL outperforms GAT due to the seasonality of bike and taxi demands. Specifically, the periodic view, comprising temporal and regional features, introduces prior knowledge that patterns repeat daily and weekly. This effectively narrows the model's search space. Additionally, the co-evolving pattern learning module addresses issues of lagged co-evolving correlation and varying co-evolving correlations throughout the day. It adaptively integrates the correlated pattern of reference demand with the target demand pattern.



(a) NYC Yellow Taxi. (b) NYC Citi Bike.

Figure. 3.11 Effect of different components.

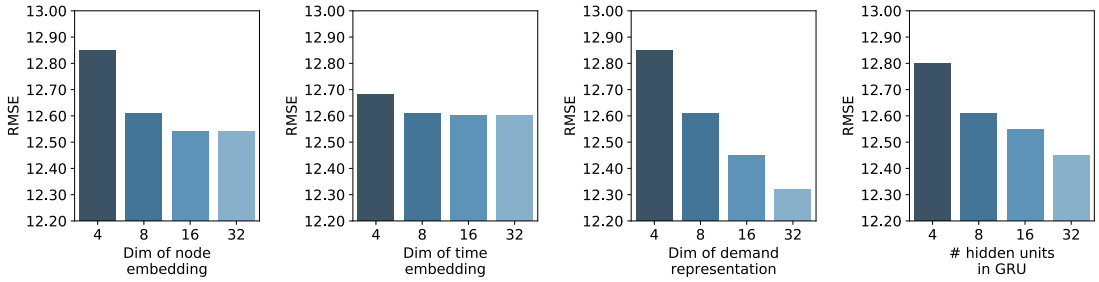


Figure. 3.12 Effect of varying settings on NYC Yellow Taxi.

3.4.6 Hyperparameter Evaluation

CEST has several hyperparameters to set manually, including the dimension of node embedding, time embedding, demand representation, and the number of GRU hidden units. We test different settings for the NYC task and present the results in Fig. 3.12 and Fig. 3.13.

Performance isn't sensitive to the dimension of time embedding in both tasks. This is because consecutive timestamps have strong correlations, placing temporal

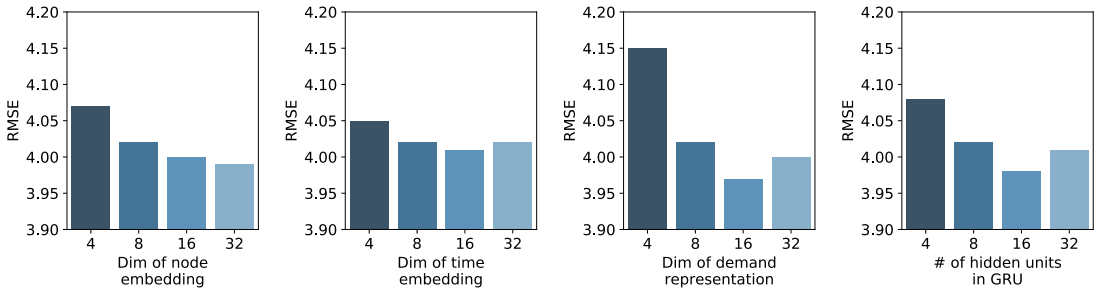
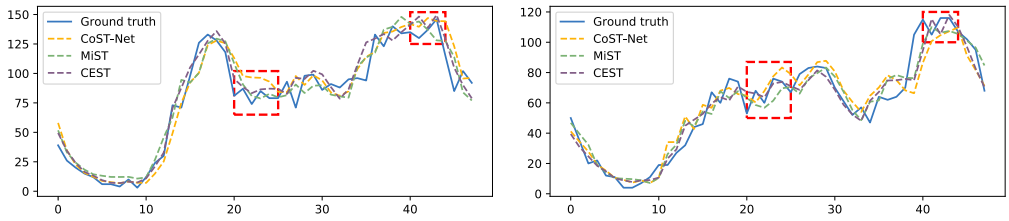


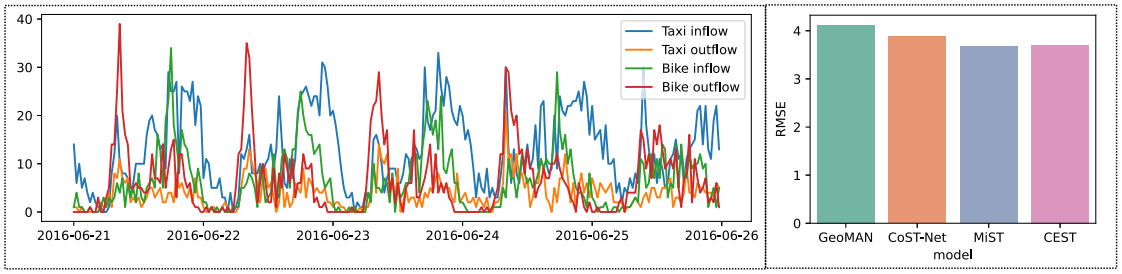
Figure. 3.13 Effect of varying settings on NYC Citi Bike.



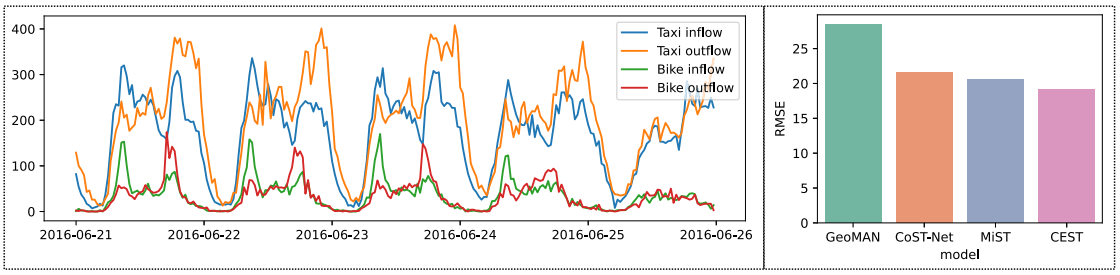
(a) Region A. (b) Region B.

Figure. 3.14 . The predicting results of our model against the ground truth and two competitive baseline models.

features in a low-dimensional latent space. Increasing the dimension of node embedding initially reduces error, with optimal results around the value of 16. This is because nodes exhibit spatial correlations, with demand evolution over neighboring nodes being similar. The structure of demand representation is more complex than node and time embeddings since it encompasses information on nodes, time, and real-time demand volume. Therefore, adjusting the dimension of demand representation significantly affects error. The number of GRU hidden units is also influential, capturing diverse temporal dynamics. Setting the last two hyperparameters too high can easily lead to overfitting.

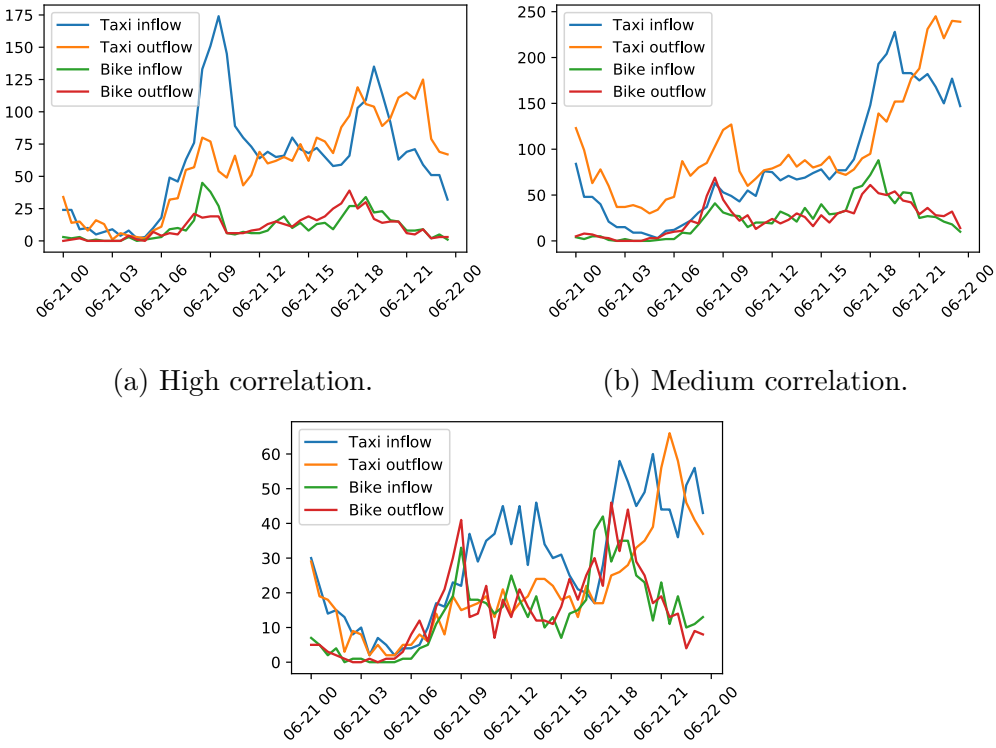


(a) Region C.



(b) Region D.

Figure. 3.15 The left part shows the movement of bike demand and taxi demand over the region, and the right part shows the performance achieved by different models.



(a) High correlation.

(b) Medium correlation.

(c) Low correlation.

Figure. 3.16 The demand evolution of three regions with high, medium and low co-evolving correlation respectively.

3.4.7 Case Study

We conduct three case studies on the NYC Yellow Taxi task to further investigate CEST's effects qualitatively.

Study on Performance at Specific Timestamps

In Fig. 3.14, we illustrate the predictions of CEST for the upcoming predictive time interval, comparing them with the ground truth and two baseline models. Some intriguing observations are highlighted with red dashed rectangles. The results show that CEST captures a more accurate local trend compared to MiST and CoST-Net.

Study on Performance Across Specific Regions

Regarding Region C, as depicted in Fig. 3.15a, CEST outperforms CoST-Net and GeoMAN but yields results comparable to MiST. It's worth noting that CEST takes into account both the periodic view and temporal shift issues, which MiST overlooks. The ineffectiveness of the periodic view in this context arises because, apart from during morning rush hours, the movements of both bike and taxi demands fluctuate drastically and irregularly, leading to a weak co-evolving correlation, even from a periodic perspective. The failure of temporal alignment can be attributed to the observed phenomenon where, in most cases, the local peak of bike demand occurs later than the corresponding peak of taxi demand. As a result, the change in taxi demand acts as a leading signal for bike demand, rendering temporal alignment ineffective.

For Region D, as shown in Fig. 3.15b, modeling the periodic view and actively aligning temporally are crucial for accurate forecasting. The effectiveness of the periodic view is due to the pronounced periodicity observed in the movements of both taxi and bike demands. Furthermore, changes in bike demand serve as leading

signals for changes in taxi demand, necessitating temporal alignment to incorporate bike data more effectively.

Study on Co-evolving Correlations

In Fig. 3.16, we display the bike and taxi demands for three regions, each with varying levels of daily average co-evolving correlation. The daily average co-evolving correlation is calculated as $\bar{\gamma} = \frac{1}{T}\sum_{t=1}^{T}\sum_{t'=t-\tau+1}^{t}\gamma_{t,t'}$, where T represents the number of time intervals in a day (i.e., 48 in our case), and $\gamma_{t,t'}$ denotes the co-evolving correlation between target demand at t and reference demand at t', as defined in Sec. 3.3.2. In Fig. 3.16a, the morning and evening peaks overlap with a slight temporal drift. Although there exists a systematic difference between taxi and bike demands during working hours, our model can discern that their underlying evolving patterns are similar. In Fig. 3.16b, post evening rush hour, bike and taxi demands exhibit diverging trends, leading to a reduced co-evolving correlation. In Fig. 3.16c, the peak of bike demand is in the morning, while the taxi demand peaks around noon, suggesting entirely different evolving patterns.

Chapter 4

Disentangling Low-Frequency Components in Spatial-Temporal Data

Time series data is characterized by its inherent complexity due to the dynamic patterns observed across spatial and temporal dimensions. Consider the determinants of traffic volume: the population density of an area, the specific time of day, the day of the week, and prevailing weather conditions. Each determinant introduces a distinct frequency component: The time of day, with its hourly variations, represents a high-frequency component; The day of the week, changing weekly, introduces a medium-frequency component; The population density of an area, which evolves over longer periods due to factors like migration, is indicative of a low-frequency component; Weather conditions, which can be erratic and not always consistent with historical traffic data, contribute an irregular component. **Collectively, these components provide insights into the space-time-varying distribution of the data.**

While many existing methodologies aim to capture these spatial-temporal correlations by integrating external spatial and temporal information, they might

not fully encapsulate the nuances of space-time-varying distribution. Given the challenge of comprehensively identifying and evaluating all relevant factors, our approach focuses on extracting these determinants directly from the time series data. These extracted determinants, which reflect the space-time-varying distribution, are subsequently integrated into our model.

Each observation in a time series dataset is influenced by multiple determinants. Isolating these determinants based on a single observation can be challenging. However, if two observations exhibit similar trends, they might be influenced by a shared determinant. For instance, if an uptick in foot traffic in one area is consistently mirrored in a neighboring area, a shared event or factor influenced by space-time-varying distribution might be the underlying cause.

In our research, we categorize these shared determinants across observations as 'low-frequency components'. Spatial attributes, for instance, serve as low-frequency components within a time series, contrasting with high-frequency components such as the time of day. To address the intricacies of space-time-varying distribution, we introduce a specialized module. This module, designed to differentiate between low and high-frequency components, is strategically incorporated at the beginning of each hidden layer in our deep learning architecture. Its integration ensures that the model captures both the broader and more specific patterns of space-time-varying distribution.

4.1 Introduction

Time series forecasting is an imperative problem in many industrial and business applications. For instance, a public transport operator can allocate sufficient capacity to mitigate the queuing time in a region in advance, if they have the means

to foresee that a particular area will suffer from a supply shortage in the next couple of hours [28]. Taking another example, an investor can avoid economic loss with the assistance of a robo-advisor which is able to predict a potential market crash [21]. Due to the complex and continuous fluctuation in impacting factors, real-world time series tends to be extraordinarily non-stationary, that is, exhibiting diverse dynamics. For instance, the traffic volume over a road is primarily affected by the road's condition, location, and current time and weather conditions. In the case of retailing, the current season, price and brand serve as determinants for merchandise sales. The diverse dynamics impose an enormous challenge on time series forecasting. In this work, we will study multi-variate time series forecasting, where multiple variables evolve with time.

Traditional time series forecasting algorithms, such as ARIMA and state space models (SSMs), provide a principled framework for modeling and learning time series patterns. However, these algorithms have a rigorous requirement for the stationarity of a time series, which encounters severe limitations in practical use if most of the impacting factors are unavailable. With the recent advance in deep learning techniques, we are now capable of handling complex dynamics as a single unit, even without any additional supplement of impacting factors. Common neural architectures applied on time series data include recurrent neural network (RNN), long-short term memory (LSTM) [39], Transformer [55], Wavenet [van den Oord et al.] and temporal convolution networks (TCN) [3].

4.1.1 Preliminary Analysis

There is a rich quantity of existing works that deal with MTS forecasting. Nonetheless, little work has identified precisely the key bottleneck in this kind of problem.

Herein, before formally proposing our solution, we start by systematically analyzing the problem to obtain greater insight. In real-world circumstances, we roughly classify an impact imposed on MTS into four classes in accordance with its activated ranges on the spatial and temporal dimensions. The four classes are composed of low-frequency local impact, low-frequency global impact, high-frequency local impact and high-frequency global impact. Here, "low-frequency" / "high-frequency" describes the activated range of the impact from the temporal view, and "global" / "local" describes the activated range from the spatial view. In particular, "low-frequency" means that the impact varies smoothly or, in other words, it tends to stay stable for a relatively long time; "high-frequency" means that the impact varies drastically; "global" means that the impact imposes a similar effect on all time series; "local" means that the impact only affects individual time series, or imposes different effects on different time series. Although the activated range on either the temporal or spatial dimension lies in a continuous spectrum, we consider only these four extreme cases as sufficient to reveal the essence of MTS. Any measurement of a time series is a mixture of four components respectively associated with the four classes of impacts, which can be formulated as follows:

$$\mathbf{X}_{i,t} = \mathbf{X}^{\mathrm{lh}}_{i,t}\mathbf{X}^{\mathrm{ll}}_{i,t}\mathbf{X}^{\mathrm{gh}}_{t}\mathbf{X}^{\mathrm{gl}}_{t} + const, \tag{4.1}$$

where $\mathbf{X}_{i,t} \in \mathbb{R}$ is the measurement of the i^{th} time series on time t, $\mathbf{X}^{\mathrm{lh}}_{i,t} \in \mathbb{R}$ denotes the local high-frequency component, $\mathbf{X}^{\mathrm{ll}}_{i,t} \in \mathbb{R}$ denotes the local low-frequency component, $\mathbf{X}^{\mathrm{gh}}_{t} \in \mathbb{R}$ denotes the global high-frequency component and $\mathbf{X}^{\mathrm{gl}}_{t} \in \mathbb{R}$ denotes the global low-frequency component. To understand this form of factorization more thoroughly, the following real-world example is used as the showcase. The time series data we present is the evolution of demand for a shared

bike over three selected regions in New York City, as manifested in Fig. 4.1. In this example, the time of day serves as a global high-frequency impact; region identity, including regional population and functionality, serves as a local low-frequency impact; the day of week serves as a global low-frequency impact. Local high-frequency impacts are indistinguishable from the raw data, such as traffic accidents or congestion.

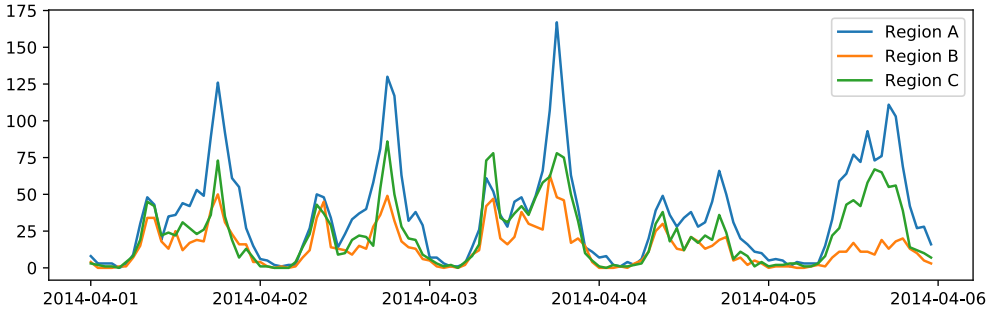


Figure. 4.1 NYC shared bike demand.

Time series forecasting is based mainly on its recent dynamics which is composed of contiguous measurements. Formally, the dynamics is expressed as a vector $[\mathbf{X}_{i,t}, \mathbf{X}_{i,t-1}, \cdots, \mathbf{X}_{i,t-\delta+1}]^\top$, where δ is the spanning time. However, canonical deep learning architectures employed in general time series forecasting tasks, such as LSTM [39], Transformer [88, 55] and Wavenet [van den Oord et al.], only capture the directional information of this vector, which is a special type of temporal relationship, that results in discarding some informative components. To obtain the specific form of temporal relationship actually modeled, we presume two postulations which hold in the majority of real-world problems: (1) the low-frequency components (including both the global low-frequency and local low-frequency components) are stable over a given period; (2) the global high-frequency component well-dominate the local high-frequency component. Based on the factorization in Eq. (4.1) along

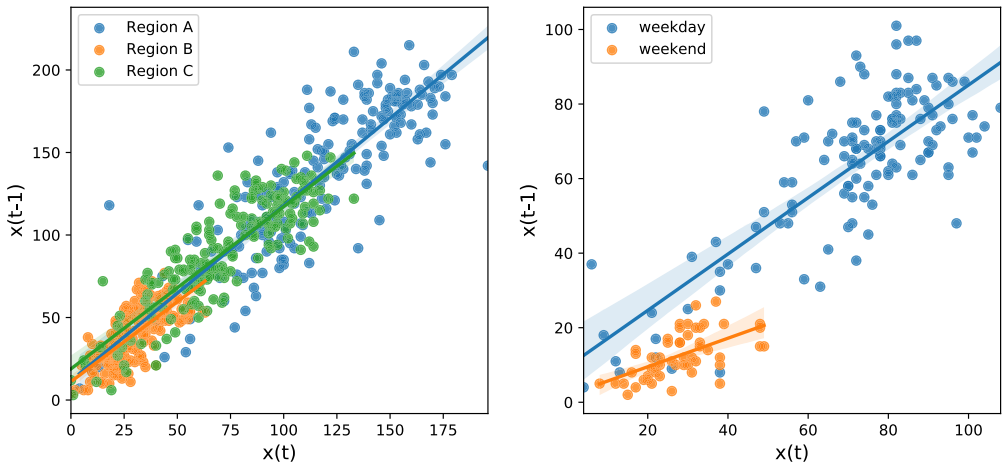
with these two postulations, it is natural to derive the direction of the vector from the basis of its constant origin, the entry associated with historical time t_0 of which is calculated as follows:

$$\begin{aligned}\frac{\mathbf{X}_{i,t_0}}{\sqrt{\sum_{t'=0}^{\delta-1}(\mathbf{X}_{i,t-t'})^2}} &= \frac{\mathbf{X}^{\text{lh}}_{i,t_0}\mathbf{X}^{\text{ll}}_{i,t_0}\mathbf{X}^{\text{gh}}_{t_0}\mathbf{X}^{\text{gl}}_{t_0}}{\sqrt{\sum_{t'=0}^{\delta-1}(\mathbf{X}^{\text{lh}}_{i,t-t'})^2(\mathbf{X}^{\text{ll}}_{i,t-t'})^2(\mathbf{X}^{\text{gh}}_{t-t'})^2(\mathbf{X}^{\text{gl}}_{t-t'})^2}} \\ &\approx \frac{\mathbf{X}^{\text{gh}}_{i,t_0}}{\sqrt{\sum_{t'=0}^{\delta-1}(\mathbf{X}^{\text{gh}}_{i,t-t'})^2}},\end{aligned} \tag{4.2}$$

where the sign of each quantity is omitted for conciseness, as they do not influence our asserted conclusion. We note that the obtained directional vector merely accounts for the global high-frequency component, totally discarding the global low-frequency component, and its local-low frequency and local high-frequency counterparts.

Discarding the other three components incurs **spatial indistinguishability** and **temporal indistinguishability**. Spatial indistinguishability means that dynamics yielded by different variables are not adequately discernible. And temporal indistinguishability means that dynamics measured at specific times are not substantially discrete. For instance, looking at the three regions in Fig. 4.1, we consider their dynamics measured between 8pm and 9pm on different days, so they share the same global high-frequency element. In Fig. 4.2a, we plot the measurement at 8pm versus the measurement at 9pm over the three regions, where the data points are colored in accordance with their regional identities. Hence, a cluster of dynamics with the same color shares the identical local low-frequency component. In Fig. 4.2b, we only plot measurement pairs of region A, and separate them based on weekday or weekend. Here, a cluster of dynamics with the same color share the identical global low-frequency component. Different clusters of dynamics are supposed to

be distinguishable, as their underlying local low-frequency components or global low-frequency components are disparate. However, the cluster-wise relationships (indicated by the direction of a straight line fitting the intra-cluster data points) are highly correlated, which signifies either the spatial or the temporal indistinguishability. Such indistinguishability hinders deep neural networks from perceiving the spatial and temporal difference. It should be noted that, once the model tunes the fraction of parameters uniquely responsible for a cluster of dynamics, the forecast of other clusters will inversely degenerate in reaction to this action. This makes the current update prone to be counteracted by any subsequent updates, as the temporary status is not even locally optimal. Hence, the ultimate model mainly captures an average property of these clusters, emanating from the common global high-frequency component. This outcome also conforms completely to our deduction in Eq. (4.2).



(a) (b)

Figure. 4.2 (a) Spatial indistinguishability; (b) Temporal indistinguishability.

4.1.2 Contributions

To address the above issues, the key is to refine more types of components from the original measurement. Thereby relationships that distinguish dynamics from the spatial view or the temporal view can be captured. In our work, we propose two kinds of normalization modules – temporal normalization (TN) and spatial normalization (SN) – which separately refine the high-frequency and local components. Specifically, the high-frequency component assists with distinguishing dynamics from the spatial view, and the local component facilitates differentiating the dynamics from the temporal view. With distinguishability on space and time, the model is able to exclusively fit each clusters of samples, especially some long-tailed samples. Moreover, we show the connection between our method and other state-of-the-art (SOTA) methods which rely on mutual relationship establishment to distinguish dynamics. We now have two prominent advantages, apart from the higher prediction accuracy: (1) the computational cost remains in $\mathcal{O}(NT)$, rather than scaling up to $\mathcal{O}(N^2T)$; (2) the converging speed is faster, as demonstrated by the experiments conducted.

4.2 Preliminaries

In this section, we introduce the definitions and the assumption. All frequently used notations are reported in Table 5.1.

Definition 4 (Time series forecasting)**.** Time series forecasting is formulated as the following conditional distribution:

$$P(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{T_{out}} P(\mathbf{Y}_{:,t}|\mathbf{X}),$$

Table 4.1 Notations

Notation	Description
N, T_{in}, T_{out}	Number of variables / input steps / output steps.
$\mathbf{X} \in \mathbb{R}^{N \times T_{in}}$	Input data.
$\mathbf{Y} \in \mathbb{R}^{N \times T_{out}}$	Output data.
$\mathbf{Z} \in \mathbb{R}^{N \times T_{in} \times d_z}$	Latent data.
$\mathbf{Z}^{\text{lh}} \in \mathbb{R}^{N \times T_{in} \times d_z}$	Local high-frequency component.
$\mathbf{Z}^{\text{ll}} \in \mathbb{R}^{N \times T_{in} \times d_z}$	Local low-frequency component.
$\mathbf{Z}^{\text{gh}} \in \mathbb{R}^{T_{in} \times d_z}$	Global high-frequency component.
$\mathbf{Z}^{\text{gl}} \in \mathbb{R}^{T_{in} \times d_z}$	Global low-frequency component.
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Vector or matrix that represents certain variable.
$+, \cdot, /$	Element-wise addition / multiplication / division.
$\overset{\text{i.i.d.}}{=}$	Two variables are i.i.d..
$*$	Placeholder.

Definition 5 (Time series factorization)**.** Time series factorization generalizes Eq. (4.1) into the latent space, which takes the following form:

$$\mathbf{Z}_{i,t} = \mathbf{Z}^{\text{lh}}_{i,t} \mathbf{Z}^{\text{ll}}_{i,t} \mathbf{Z}^{\text{gh}}_{t} \mathbf{Z}^{\text{gl}}_{t}, \tag{4.3}$$

where:

$$\mathbf{Z}^{\text{lh}}_{i,t} \overset{\text{i.i.d.}}{=} \mathbf{Z}^{\text{lh}}_{i,t-1},\ \mathbf{Z}^{\text{ll}}_{i,t} \approx \mathbf{Z}^{\text{ll}}_{i,t-1},\ \mathbf{Z}^{\text{gh}}_{t} \overset{\text{i.i.d.}}{=} \mathbf{Z}^{\text{gh}}_{t-1},\ \mathbf{Z}^{\text{gl}}_{t} \approx \mathbf{Z}^{\text{gl}}_{t-1},\ \mathbf{Z}^{\text{l*}}_{i,t} \overset{\text{i.i.d.}}{=} \mathbf{Z}^{\text{l*}}_{j,t}.$$

Assumption 1. The set of elements of $\mathbf{Z}^{\text{lh}}_{i,t}$, $\mathbf{Z}^{\text{ll}}_{i,t}$, $\mathbf{Z}^{\text{gh}}_{t}$ and $\mathbf{Z}^{\text{gl}}_{t}$ are mutually independent, which is formally written as:

$$P(\mathbf{Z}^{\text{ll}}_{i,t}, \mathbf{Z}^{\text{lh}}_{i,t}, \mathbf{Z}^{\text{gh}}_{t}, \mathbf{Z}^{\text{gl}}_{t}) = \prod_{k=1}^{d_z} P(\mathbf{Z}^{\text{ll}}_{i,t,k}) P(\mathbf{Z}^{\text{lh}}_{i,t,k}) P(\mathbf{Z}^{\text{gh}}_{t,k}) P(\mathbf{Z}^{\text{gl}}_{t,k}). \tag{4.4}$$

4.3 Methodology

We display the overview of the architecture being leveraged in our work in Fig. 5.3. Some key variables with their shapes are labeled at their corresponding

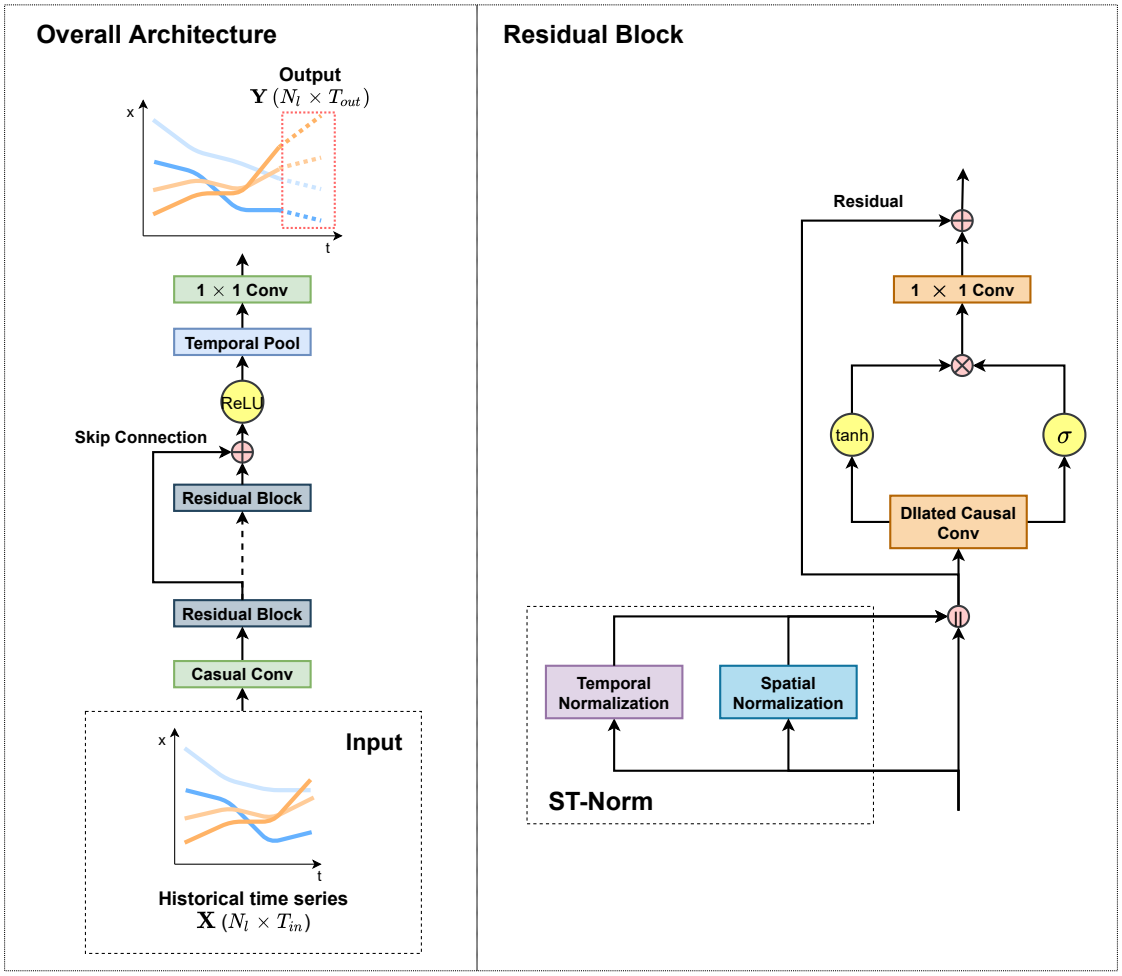


Figure. 4.3 Overall architecture, where we just draw two residual blocks for illustration, but multiple blocks can be stacked layer by layer. $+$, $\times$ and $||$ respectively denote element-wise addition, element-wise multiplication and concatenation

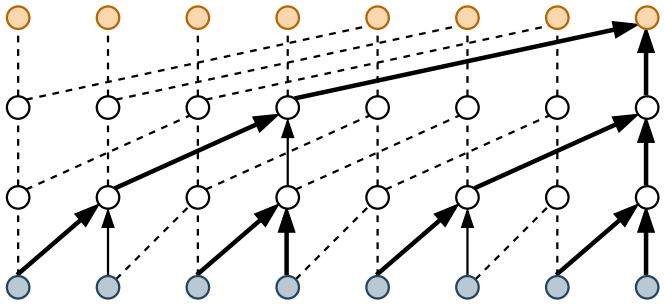


Figure. 4.4 Dilated Causal Convolution.

positions along the computation path. Generally, our framework follows a structure similar to Wavenet[3], except that we add both spatial normalization and temporal normalization modules which together are abbreviated as ST-Norm or STN.

4.3.1 Dilated Causal Convolution

In this section, we briefly introduce dilated causal convolution where the filter is applied with skipping values. For a 1-D signal $\mathbf{z} \in \mathbb{R}^T$ and a filter $f : \{0, \ldots, k-1\} \rightarrow \mathbb{R}$, the causal convolution on element t is defined as follows:

$$F(t) = (\mathbf{z} * f)(t) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{z}_{t-i}. \tag{4.5}$$

This formula can be easily generalized for multi-dimension signal but we omit its general form here for brevity. Moreover, padding (zero or replicate) with size of $k-1$ is appended to the left tail of the signal to ensure length consistency. We can stack multiple causal convolution layers to obtain a larger receptive field for each element.

One shortcoming of using causal convolution is that either the kernel size or the number of layers increases in a linear manner with the range of the receptive field, and the linear relationship causes an explosion of parameters when modeling

long history. Pooling is a natural choice to address this issue, but it sacrifices the order information presented in the signal. To this end, dilated causal convolution is leveraged, a form which supports the exponential expansion of the receptive field. The formal computing process is written as:

$$F(t) = (\mathbf{z} *_d f)(t) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{z}_{t-d \cdot i}, \tag{4.6}$$

where d is the dilation factor. Normally, d increases exponentially w.r.t. the depth of the network (i.e., 2^l at level l of the network). If d is 1 (2^0), then the dilated convolution operator $*_d$ reduces to a regular convolution operator $*$.

4.3.2 Temporal Normalization

Temporal normalization (TN) aims to refine the high-frequency components – both global and local – from the hybrid signal. Here, for conciseness, we introduce two notations to individually summarize high-frequency components and low-frequency components, which are expressed as:

$$\mathbf{Z}_{i,t}^{\text{high}} = \mathbf{Z}_{i,t}^{\text{lh}} \mathbf{Z}_t^{\text{gh}}, \ \mathbf{Z}_{i,t}^{\text{low}} = \mathbf{Z}_{i,t}^{\text{ll}} \mathbf{Z}_t^{\text{gl}}.$$

The applicability of TN is based on a reasonable assumption that the changing rates of low-frequency components are much slower than those of the high-frequency components. Or more technically, each low-frequency component approximately equals a constant over a period. Under this assumption, we are can apply TN on time series without the additional supplement of features characterizing the frequency. Such characteristic is well-suitable for an ample number of real-world problems, where the specific frequency is not available.

We start with expanding $\mathbf{Z}_{i,t}^{\text{high}}$ to obtain a desirable form whose distinct quantities can be derived from data:

$$\begin{aligned}
\mathbf{Z}_{i,t}^{\text{high}} &= \frac{\mathbf{Z}_{i,t}^{\text{high}} - E\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right)}{\sigma\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right)+\epsilon}\sigma\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right) + E\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right) \\
&= \frac{\mathbf{Z}_{i,t}^{\text{high}}\mathbf{Z}_{i,t}^{\text{low}} - \mathbf{Z}_{i,t}^{\text{low}} E\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right)}{\mathbf{Z}_{i,t}^{\text{low}}\sigma\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right)+\epsilon}\sigma\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right) + E\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right) \\
&= \frac{\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right)}{(\pm)\sigma\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right)+\epsilon}\sigma\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right) + E\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right) \\
&= \frac{\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right)}{\sigma\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right)+\epsilon}\left((\pm)\sigma\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right)\right) + E\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right),
\end{aligned} \tag{4.7}$$

where ϵ is a small constant to preserve numerical stability; $\mathbf{Z}_{i,t}$ is observable; $E\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right)$ and $(\pm)\sigma\left(\mathbf{Z}_{i,t}^{\text{high}}\middle|i\right)$ are mean and standard deviation (plus or minus) of the high-frequency impact on the i^{th} time series over time, which can be approximated by a pair of learnable vectors γ_i^{high} and β_i^{high} with the size of d_z. To estimate $E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right)$ and $\sigma\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right)$ can be estimated as follows under Def. 5 and Assumption 1:

$$\begin{aligned}
E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right) &\approx \frac{1}{\delta}\sum_{t'=1}^{\delta}\mathbf{Z}_{i,t-t'+1}^{\text{high}}\mathbf{Z}_{i,t}^{\text{low}} \\
&\approx \frac{1}{\delta}\sum_{t'=1}^{\delta}\mathbf{Z}_{i,t-t'+1}^{\text{high}}\mathbf{Z}_{i,t-t'+1}^{\text{low}} \\
&= \frac{1}{\delta}\sum_{t'=1}^{\delta}\mathbf{Z}_{i,t-t'+1} \\
\sigma^2\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right) &= E\left[\left(\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right)\right)^2\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right] \\
&\approx \frac{1}{\delta}\sum_{t'=1}^{\delta}\left(\mathbf{Z}_{i,t-t'+1}^{\text{high}}\mathbf{Z}_{i,t}^{\text{low}} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{i,t}^{\text{low}},i\right)\right)^2
\end{aligned} \tag{4.8}$$

$$\approx \frac{1}{\delta}\sum_{t'=1}^{\delta}\left(\mathbf{Z}^{\text{high}}_{i,t-t'+1}\mathbf{Z}^{\text{low}}_{i,t-t'+1} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}^{\text{low}}_{i,t}, i\right)\right)^2$$
$$= \frac{1}{\delta}\sum_{t'=1}^{\delta}\left(\mathbf{Z}_{i,t-t'+1} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}^{\text{low}}_{i,t}, i\right)\right)^2, \tag{4.9}$$

where δ is a period during which the low-frequency component approximately remains to be a constant. In our work, for simplicity, we let δ equal to the number of input time steps. By substituting the estimations of the four unobservable variables into Eq. (4.7), we are able to obtain the representation of the high-frequency component:

$$\mathbf{Z}^{\text{high}}_{i,t} = \frac{\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}^{\text{low}}_{i,t}, i\right)}{\sigma\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}^{\text{low}}_{i,t}, i\right) + \epsilon}\gamma^{\text{high}}_{i} + \beta^{\text{high}}_{i} \tag{4.10}$$

Noticeably, TN has a close relationship with instance normalization (IN) for image data [86], where style plays the role of a low-frequency component and content serves as a high-frequency component. The novelty of our work is that we trace the origin of TN under the context of MTS, and deduce TN step-by-step from its origin.

4.3.3 Spatial Normalization

The objective of spatial normalization (SN) is to refine local components, composed of the local high-frequency component and the local low-frequency component. To achieve this objective, the primary task is first to eliminate global components, resulted from global impacts such as time of day, day of week and weather condition, etc. We also introduce two notations to summarize local and global components:

$$\mathbf{Z}^{\text{global}}_{t} = \mathbf{Z}^{\text{gh}}_{t}\mathbf{Z}^{\text{gl}}_{t},\ \mathbf{Z}^{\text{local}}_{i,t} = \mathbf{Z}^{\text{lh}}_{i,t}\mathbf{Z}^{\text{ll}}_{i,t}.$$

Likewise, the applicability of SN is based on the assumption that the global impacts impose similar effects on all time series. For instance, in Fig. 4.1, there are common upward trends over the three regions with similar increasing rates when moving from 8am to 9am. Here, we need to clarify that we do not require the global impacts to strictly exert the same influence on each time series. Those effects that are not evenly observed on each time series could be complemented by the defined local component.

We firstly expand $\mathbf{Z}_{i,t}^{\text{local}}$ to an expression where each term can be approximated from data or be assigned with learnable parameters:

$$\begin{aligned}
\mathbf{Z}_{i,t}^{\text{local}} &= \frac{\mathbf{Z}_{i,t}^{\text{local}} - E\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right)}{\sigma\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right)+\epsilon}\sigma\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right) + E\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right) \\
&= \frac{\mathbf{Z}_{i,t}^{\text{local}}\mathbf{Z}_{t}^{\text{global}} - \mathbf{Z}_{t}^{\text{global}}E\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right)}{\mathbf{Z}_{t}^{\text{global}}\sigma\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right)+\epsilon}\sigma\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right) + E\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right) \\
&= \frac{\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{t}^{\text{global}},t\right)}{(\pm)\sigma\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{t}^{\text{global}},t\right)+\epsilon}\sigma\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right) + E\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right) \\
&= \frac{\mathbf{Z}_{i,t} - E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{t}^{\text{global}},t\right)}{\sigma\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{t}^{\text{global}},t\right)+\epsilon}\left((\pm)\sigma\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right)\right) + E\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right)
\end{aligned} \tag{4.11}$$

where $\mathbf{Z}_{i,t}$ is directly observable; $(\pm)\sigma\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right)$ and $E\left(\mathbf{Z}_{i,t}^{\text{local}}\middle|t\right)$ are approximated by two learnable vectors[1] γ^{local} and β^{local}; the estimation of $E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{t}^{\text{global}},t\right)$ and $\sigma\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_{t}^{\text{global}},t\right)$ can be derived from data in the following ways under Def. 5 and

[1]Each time would possess the identical prior distribution if dynamic laws, such as periodicity, is unknown.

Assumption 1:

$$
\begin{aligned}
E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_t^{\text{global}},t\right) &\approx \frac{1}{N}\sum_{j=1}^{N}\mathbf{Z}_{j,t}^{\text{local}}\mathbf{Z}_t^{\text{global}} \\
&= \frac{1}{N}\sum_{j=1}^{N}\mathbf{Z}_{j,t} \qquad (4.12)\\
\sigma^2\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_t^{\text{global}},t\right) &= E\left[\left(\mathbf{Z}_{i,t}-E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_t^{\text{global}},t\right)\right)^2\middle|\mathbf{Z}_t^{\text{global}},t\right] \\
&\approx \frac{1}{N}\sum_{j=1}^{N}\left(\mathbf{Z}_{j,t}^{\text{local}}\mathbf{Z}_t^{\text{global}}-E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_t^{\text{global}},t\right)\right)^2 \\
&= \frac{1}{N}\sum_{j=1}^{N}\left(\mathbf{Z}_{j,t}-E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_t^{\text{global}},t\right)\right)^2 \qquad (4.13)
\end{aligned}
$$

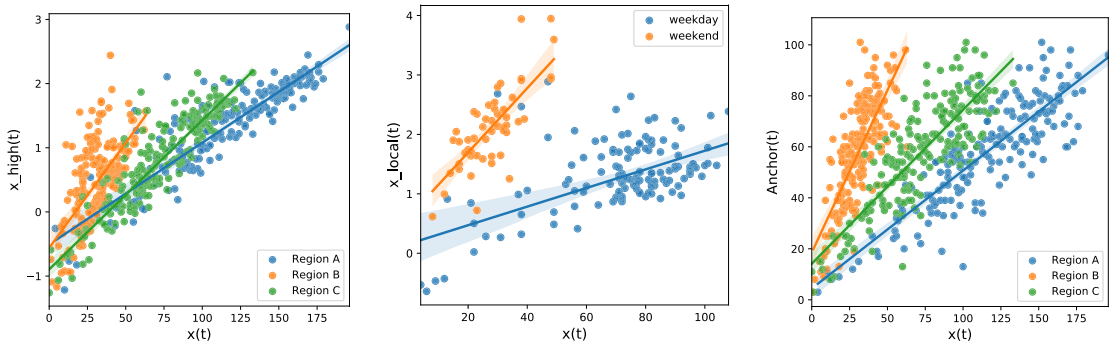
By substituting the approximations of the four unobservable variables into Eq. (4.11), we are able to obtain the composite representation of the local components:

$$
\mathbf{Z}_{i,t}^{\text{local}} = \frac{\mathbf{Z}_{i,t}-E\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_t^{\text{global}},t\right)}{\sigma\left(\mathbf{Z}_{i,t}\middle|\mathbf{Z}_t^{\text{global}},t\right)+\epsilon}\gamma^{\text{local}}+\beta^{\text{local}} \qquad (4.14)
$$

SN is a counterpart of TN in the spatial domain, where high-frequency components act as local components, and low-frequency components correspond to global components. By distilling the local or high-frequency components from the original signal, the model can capture fine-grained variation, which is extraordinarily instrumental in time series forecasting.

4.3.4 Forecasting and Learning

We let $\mathbf{Z}^{(L)} \in \mathbb{R}^{N_l \times T_{in} \times d_z}$ denote the output from the last residual block, where each row $\mathbf{z}^{(L)} \in \mathbb{R}^{T_{in} \times d_z}$ represents a variable. Then, we employ a temporal pooling block to perform temporal aggregation for each variable. Several types of pooling operations can be applied, such as max pooling and mean pooling, depending on the problem being studied. In our case, we select the vector in the most recent time



(a) TN (b) SN (c) Graph-based

Figure. 4.5 Relationships produced by the three operations.

slot as the pooling result, which is treated as the representation of the entire signal. Finally, we make a separate prediction for each variable, based on the obtained representation by a shared fully connected layer.

In the learning phase, our objective is to minimize the mean squared error between the predicted values and ground truth values. In addition, we use the Adam optimizer [46] to optimize this target.

4.3.5 Discussion

4.3.5.1 Feature Space

To illustrate how TN and SN reframe the feature space, we apply them over the raw input data, and examine whether they mitigate the issues we raise in Fig. 4.2. We plot the original quantity versus the temporally normalized quantity in Fig. 4.5a, and the original quantity versus the spatially normalized quantity in Fig. 4.5b. It is apparent that the pairwise relationship between the original quantity and the temporally normalized quantity separates different regions, and the pairwise

relationship between the original quantity and the spatially normalized quantity separates different days.

4.3.5.2 Computational Complexity

A few SOTA approaches [104, 103, 2] propose to establish a mutual relationship between different time series in order to refine the local component. In essence, they contrast a pair of time series which share the same global components over time, thereby allowing the local component of individual time series to be highlighted. For instance, we contrast the three time series in Fig. 4.1 with a single time series (regarded as an anchor), which results in a pairwise relationship reflecting the identity of each time series as shown in Fig. 4.5c. However, the eligible anchors are often unknown, and different time series may need to be paired with different anchors. To automatically identify the anchor for each time series, these methods employ a graph-learning module to explore every possible pair of time series. Their computational complexity is $\mathcal{O}(TN^2)$. Unlike other approaches proposed in this area, the normalization modules involved in our method only require $\mathcal{O}(TN)$ operations.

4.3.5.3 Generalizability

ST-Norm's efficacy guarantee hinges on the behavior consistency of time series between training data and testing data, which is applicable for scenarios with demonstration of stable and recurring patterns, where low-frequency components are subject to little or slight changes. Specifically, behavior consistency ensures the unbiasedness of the derived mean and standard deviation for testing data, effectively centering the group of normalized features at 0, averting the occurrence of distribution shift from the training data. As such, ST-Norm is expected to deliver

generalizablility, provided that the distribution of testing data does not deviate substantially from that of the training data. For adjustment with respect to data structuring, such as modifying the spatial resolution or the temporal granularity, the training data and testing data will be impacted in a consistent manner, unless using different adjustment configurations for them, respectively, so the divergence will be maintained at distribution level. For complex scenarios undergoing unpredictable variation in low-frequency components, our next study has the potential to cope with.

4.4 Evaluation

In this section, we conduct extensive experiments on three common datasets to validate the effectiveness of ST-Norm from different aspects.

4.4.1 Experimental Setting

4.4.1.1 Datasets

We validate our model on three real-world datasets, including BikeNYC, PeMSD7 and Electricity. The statistics regarding each dataset as well as the corresponding settings of the designed task are reported in Table 5.2. We standardize the values in each dataset to facilitate training and transform them back to the original scale in the testing phase.

In Table 5.2, statistics of the datasets are reported. More details regarding the datasets are introduced below.

- PeMSD7 [113]. The data is collected from Caltrans Performance Measurement System (PeMS) by sensor stations, which are deployed to monitor traffic speed

Table 4.2 Dataset statistics.

Tasks	Electricity	PeMSD7	BikeNYC
Start time	10/1/2014	5/1/2012	4/1/2014
End time	12/31/2014	6/30/2012	9/30/2014
Sample rate	1 hour	30 minutes	1 hour
# Timesteps	2184	2112	4392
# Variate	336	228	128
Training size	1848	1632	3912
Validation size	168	240	240
Testing size	168	240	240
Output length	3	3	3

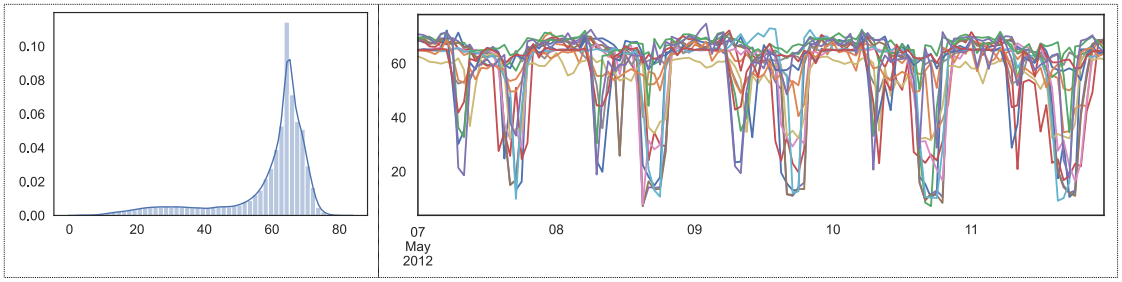
across the major metropolitan areas of the California state highway system. We further aggregate the data to 30-minute interval by average pooling.

- Electricity[2]. The original dataset contains the electricity consumption of 370 points/clients, from which 34 outlier points that contain extreme values are removed. Moreover, we calculate the hourly average consumption for each point, and take it as the time series being modeled.

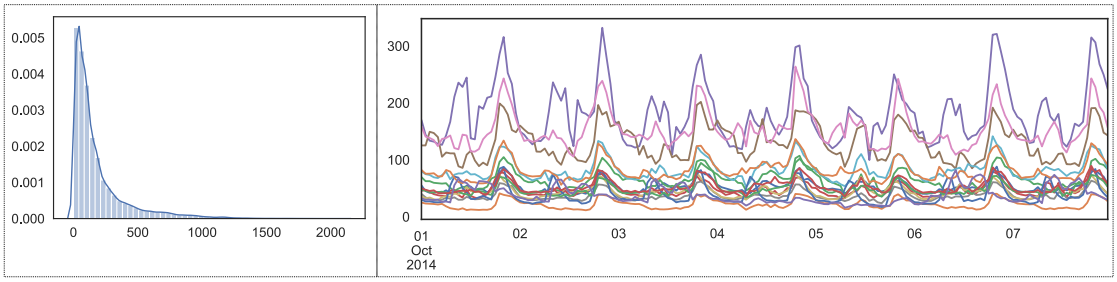
- BikeNYC [117]. Each time series in this dataset denotes the aggregate demand for shared bikes over a region in New York City. We do not consider the spatial relationship presented in the PeMSD7 and BikeNYC data, since our objective is to study the temporal patterns.

Furthermore, we display the data distribution and several exemplar time series in Fig. 5.8 to gain more insights from each dataset. We can observe that each of the three types of data lays in a wide range of scale, and exhibits periodicity to some extent. However, their evolving patterns are entirely different where the electricity time series shows the greatest diversity.

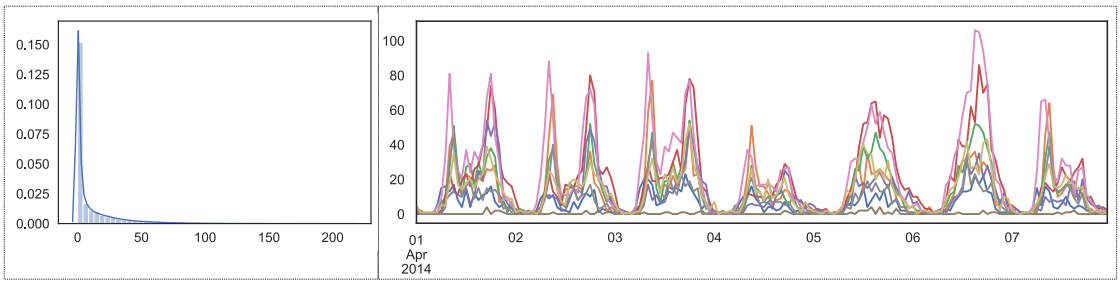
[2]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014



(a) PeMSD7.



(b) Electricity.



(c) BikeNYC.

Figure. 4.6 For each (a), (b) and (c), the left figure shows the probability density function of observed values aggregated from all variables at all time steps, and the right figure displays some sample time series.

4.4.1.2 Network Setting

We add an instance normalization (IN) module [86] in parallel with SN and TN as another complement[3]. The batch size is 4, and the input length of the batch sample is 16. For the Wavenet backbone, the layer number is set to 4, the kernel size of each DCC component is 2, and the associated dilation rate is 2^i, where i is the index of the layer (counting from 0). Such settings collectively enable the output from Wavenet to perceive 16 input steps. The number of hidden channels d_z in each DCC is 16. We apply zero-padding on the left tail of the input to enable the length of the output from DCC to equal to 16 as well. The learning rate of the Adam optimizer is 0.0001.

4.4.1.3 Evaluation Metrics

We validate our model by root mean squared error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE). We repeat the experiment ten times for each model on each dataset and report the mean of the results.

4.4.2 Baseline Models

- **MTGNN** [103]. MTGNN constructs inter-variate relationships by introducing a graph-learning module. Specifically, the graph learning module connects each hub node with its top k nearest neighbors in a defined metric space. MTGNN's backbone architecture for temporal modeling is Wavenet.

- **Graph Wavenet** [104]. The architecture of Graph Wavenet is like MTGNN. The major difference is that the former derives a soft graph where each pair of nodes has a continuous probability of being connected.

[3]The implementation can be found in our code.

- **AGCRN** [2]. AGCRN also equips with a graph-learning module to establish inter-variate relationship. Furthermore, it uses a personalized RNN to model the temporal relationship for each time series.

- **Transformer** [55]. This model captures the long-term dependencies in time series data through using an attention mechanism, where the keys and queries are yielded by causal convolution over local context to model segment-level correlation.

- **LSTNet** [49]. There are two components in LSTNet: one is a conventional autoregressive model, and the other is an LSTM with an additional skip connection over the temporal dimension.

- **TCN**. [3] The architecture of TCN is like Wavenet, except that the nonlinear transformation in each residual block is made up of two rectified linear units (ReLU).

We also test the performance of TCN and Transformer incorporating STN, where STN is similarly applied before the causal convolution operation in each layer.

4.4.3 Experiment Results

Table 4.3 Performance on the BikeNYC dataset

Models	1 hour			2 hour			3 hour		
	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE
LSTNet	19.6%	2.55	5.35	21.1%	2.77	6.04	22.6%	2.99	6.63
AGCRN	17.3%	2.34	4.76	18.7%	2.56	5.51	20.3%	2.77	6.07
Graph Wavenet	18.0%	2.39	4.78	19.4%	2.65	5.53	20.8%	2.86	6.05
MTGNN	19.0%	2.55	5.05	21.1%	2.88	6.00	22.9%	3.13	6.61
Transformer	22.8%	2.98	6.16	27.1%	3.66	7.88	29.7%	4.12	8.95
Transformer + STN	17.7%	2.36	4.75	19.1%	2.57	5.51	20.7%	2.78	6.09
TCN	22.4%	2.90	5.98	26.4%	3.57	7.66	29.1%	4.07	8.76
TCN + STN	**16.8%**	2.30	4.51	18.6%	2.55	5.34	20.4%	2.77	5.92
Wavenet	22.1%	2.86	5.92	26.3%	3.52	7.58	29.0%	3.97	8.68
Wavenet + STN	16.9%	**2.23**	**4.48**	**18.3%**	**2.47**	**5.28**	**20.1%**	**2.68**	**5.88**
Improvements	+2.8%	+4.7%	+5.8%	+2.1%	+3.5%	+4.1%	+0.9%	+3.2%	+2.8%

Table 4.4 Performance on the PeMSD7 dataset.

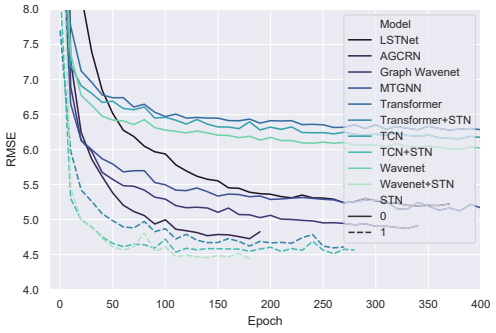
Models	30 min			60 min			90 min		
	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE
LSTNet	8.10%	3.88	6.52	8.43%	4.01	6.71	9.06%	4.30	7.12
AGCRN	4.87%	2.34	4.24	6.48%	3.07	5.58	7.27%	3.43	6.19
Graph Wavenet	4.90%	2.33	4.25	6.77%	3.19	5.69	7.57%	3.57	6.25
MTGNN	5.18%	2.46	4.48	7.61%	3.57	6.31	9.03%	4.25	7.26
Transformer	5.82%	2.75	5.03	9.31%	4.34	7.51	11.8%	5.49	9.02
Transformer + STN	4.86%	2.33	4.26	6.50%	3.08	5.65	7.33%	3.48	6.31
TCN	5.80%	2.75	4.97	9.44%	4.43	7.53	12.0%	5.61	9.06
TCN + STN	4.91%	2.34	4.22	6.42%	3.04	5.51	7.12%	3.38	6.05
Wavenet	5.50%	2.61	4.80	8.75%	4.10	7.20	11.0%	5.16	8.61
Wavenet + STN	**4.71%**	**2.25**	**4.12**	**6.23%**	**2.95**	**5.48**	**6.97%**	**3.29**	**6.01**
Improvements	+3.2%	+3.4%	+3.0%	+3.8%	+3.9%	+1.7%	+4.1%	+4.0%	+2.9%

Table 4.5 Performance on the Electricity dataset.

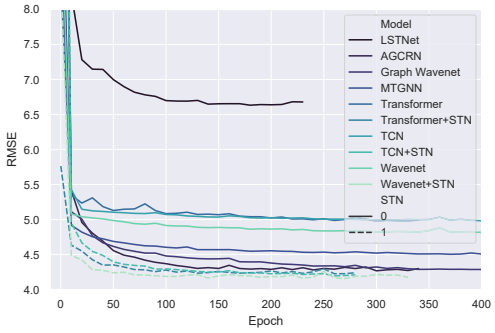
Models	1 hour			2 hour			3 hour		
	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE
LSTNet	22.4%	31.1	61.2	23.0%	31.8	62.6	24.8%	33.8	66.8
AGCRN	12.2%	17.1	36.3	16.1%	22.3	49.1	19.1%	26.0	56.6
Graph Wavenet	10.9%	15.9	34.9	15.8%	22.4	49.9	19.1%	26.5	57.7
MTGNN	11.1%	15.8	32.5	16.0%	22.2	46.3	19.6%	26.5	54.6
Transformer	11.2%	16.6	36.3	17.6%	25.4	53.7	22.2%	31.8	65.0
Transformer + STN	13.2%	17.4	35.9	17.7%	23.5	48.8	21.2%	27.9	58.0
TCN	11.1%	16.3	35.5	17.3%	25.0	52.4	21.5%	30.7	62.0
TCN + STN	13.2%	16.7	31.7	16.5%	21.5	42.8	20.1%	25.4	50.9
Wavenet	**10.8%**	15.8	33.3	16.8%	23.8	49.5	21.1%	29.5	60.3
Wavenet + STN	12.0%	**15.6**	**30.9**	**15.1%**	**20.1**	**42.2**	**17.1%**	**23.0**	**49.2**
Improvements	-11.0%	+1.2%	+4.9%	+4.4%	+9.4%	+8.8%	+10.4%	+11.5%	+9.8%

The experimental results on the BikeNYC, PeMSD7 and Electricity datasets are separately reported in Table 5.3, Table 5.4 and Table 5.5. The improvements achieved by Wavenet + STN over the best benchmarks are recorded in the last row of each table.

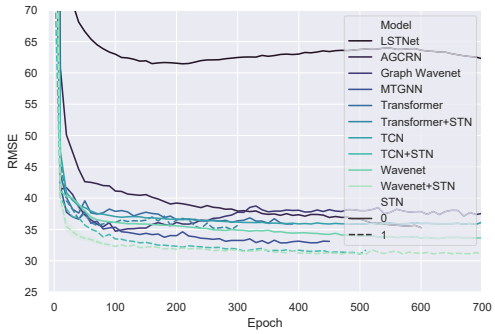
It is obvious that Wavenet + STN achieves SOTA results over almost all horizons on BikeNYC, PeMSD7 and Electricity data. The reason is that we refine the high-frequency components from both the temporal view and the spatial view, which are generally overlooked by baseline models. Next, we reveal the cause of Wavenet + STN's under-performance on the electricity dataset over the first horizon with respect to MAPE. As shown in Fig. 4.6b, electricity data follows a long-tailed distribution – there is a certain portion of quantities exceeding a relatively high level. Recall that the optimization targets minimizing mean squared error, which means that more weights are placed on large errors. Moreover, every sample is



(a) BikeNYC



(b) PeMSD7



(c) Electricity

Figure. 4.7 Loss convergence.

treated equivalently in the estimation of global statistics. Therefore, the model can fit long-tailed samples better, but at the cost of degrading the fitness on normal samples.

We also display the process of loss convergence in Appendix **??**. It shows that with the additional STN module, the converging speeds of the models are accelerated by a large margin, faster than that of nearly all baseline models.

4.4.4 Ablation Study

To validate the effectiveness of SN and TN, we design several variants as follows. We also investigate whether a graph-learning module complements STN by testing

Table 4.6 Ablation Study

		GSTN	STN	Graph	TN	SN	Vanilla
B	RMSE	**5.18**	5.21	5,46	5.63	6.37	7.40
	MAE	**2.45**	2.47	2.63	2.64	2.99	3.44
	MAPE	**18.4%**	18.7%	19.4%	19.8%	22.6%	25.8%
P	RMSE	**5.16**	5.22	5.40	5.35	6.12	6.87
	MAE	**2.77**	2.83	3.03	2.90	3.47	3.96
	MAPE	**5.86%**	5.97%	6.41%	6.08%	7.41%	8.43%
E	RMSE	**38.9**	40.8	47.5	44.1	45.9	47.9
	MAE	**18.9**	19.6	21.6	21.4	22.6	23.1
	MAPE	**14.4%**	14.7%	15.3%	16.2%	16.7%	15.9%

a variant containing them both. As all the variants contain the standard Wavenet backbone, we omit Wavenet in the name for brevity.

- **GSTN**. STN with an adaptive graph learning module as in Graph Wavenet.
- **Graph**. Graph Wavenet.
- **SN**. Wavenet with SN module.
- **TN**. Wavenet with TN module.

We evaluate these variants on all the three datasets and report the overall results in Table 5.6. It is evident that both of SN and TN contribute to the enhancement. Moreover, with an adaptive graph-learning module, the performance of STN rises marginally. We can conclude that STN largely substitutes and surpasses the graph-learning module.

4.4.5 Hyper-parameter Analysis

We further study the effect of different settings of the hyper-parameters in the proposed modules. There are four hyper-parameters to be manually set by practitioners,

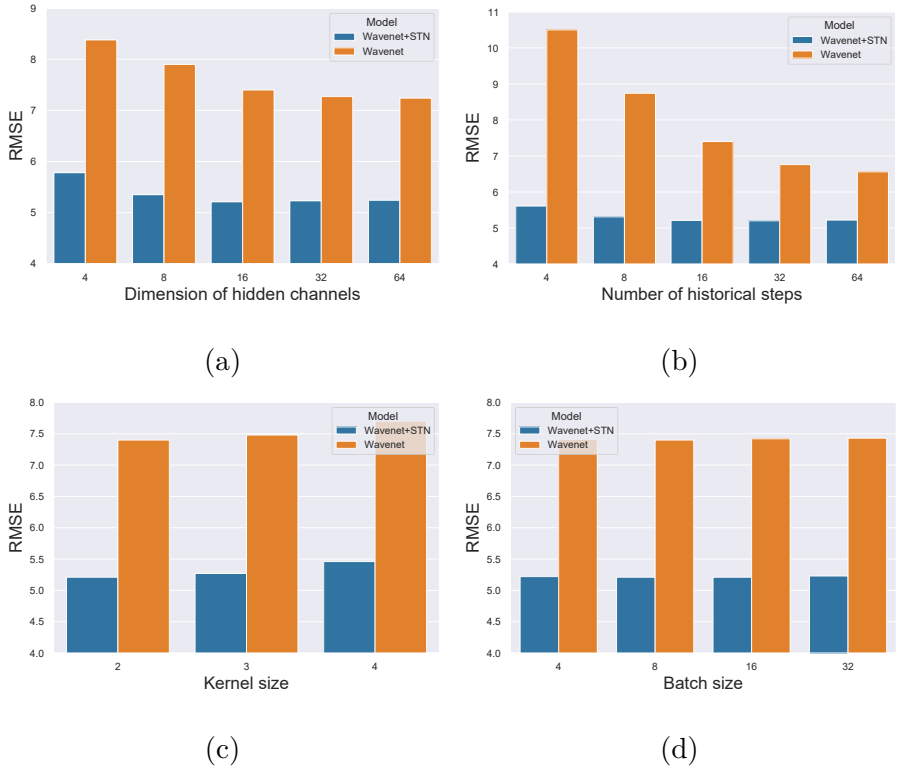


Figure. 4.8 Hyper-parameter analysis.

consisting of the dimension of hidden channels d_z, the number of historical steps input to the model, the kernel size of DCC and the batch size. The study results are reported in Fig. 5.9, from which we are able to draw a major conclusion: STN not only boosts the performance, but also increases the stability of the performance under different hyper-parameter settings.

4.4.6 Case Study

The time series data we leveraged for the case study is BikeNYC. For each of SN and TN, we examine three representative regions at specified times to reflect what the module extracts from data. We collect the intermediate representations output from the two normalization modules installed in the top residual block and

compress them via t-SNE for the sake of visualization. For comparison, we also examine their associated input representations, each of which is a concatenation of raw measurements. Next, we will discuss separately the outcomes of the two modules in details.

4.4.6.1 Spatial Normalization

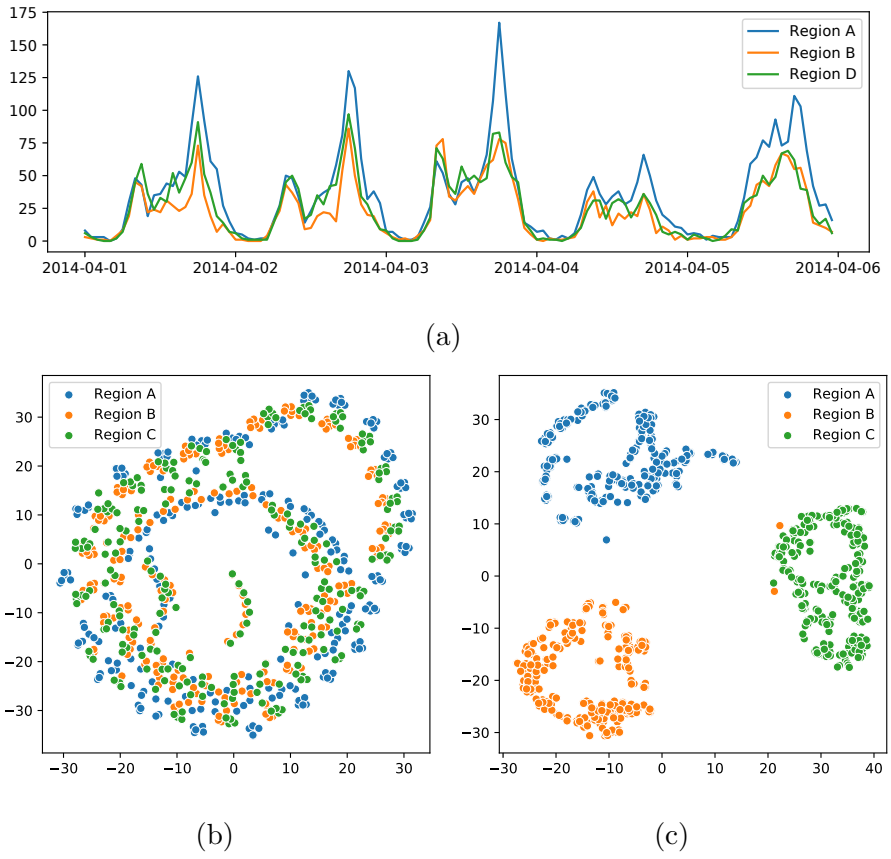


Figure. 4.9 Case study on SN.

SN removes the global component $\mathbf{Z}_t^{\text{global}}$ from the original measurement, while retaining the local component $\mathbf{Z}_{i,t}^{\text{local}}$. In Fig. 4.9a, we display the demand evolution during a given period over the three investigated regions. We can observe that the three regions have similar evolution patterns, especially regions B and C. The

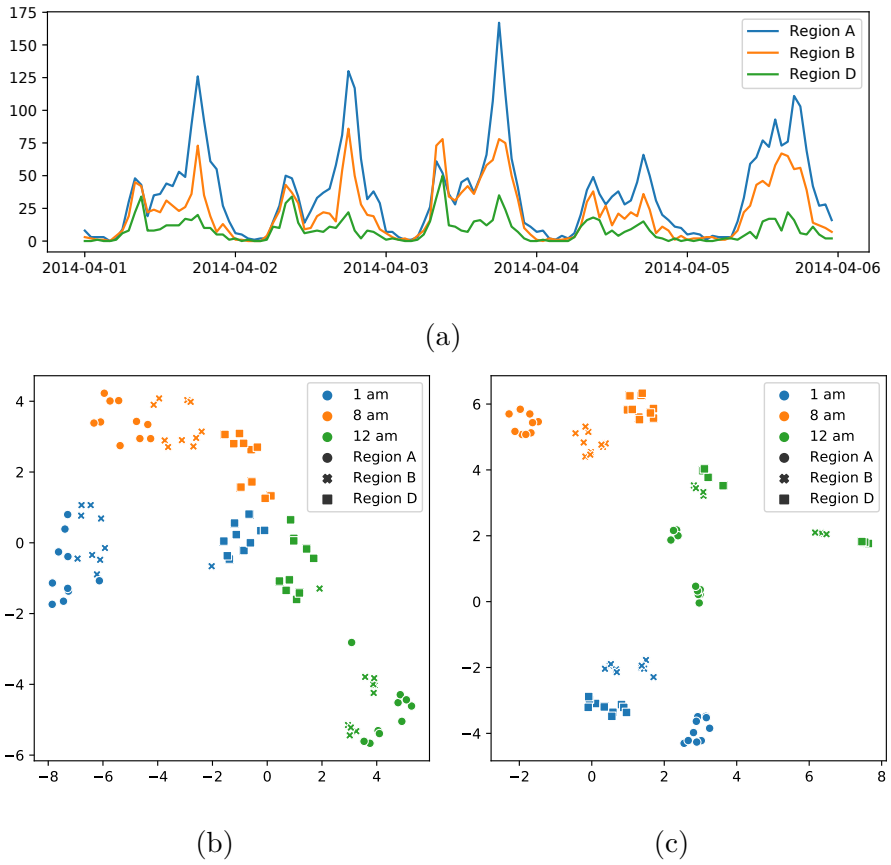


Figure. 4.10 Case study on TN.

representations concatenated by original measurements are plotted in Fig. 4.9b, and the intermediate representations output from SN are plotted in Fig. 4.9c. We can observe that SN completely rearranges the representations in accordance with its regional identity. This observation demonstrates that the low-frequency parts in the local components are roughly invariant within the group belonging to the same region. This coincides with our understanding that some regional attributes, such as the population and the functionality, are stable over time.

4.4.6.2 Temporal Normalization

TN attempts to eliminate the low-frequency component $\mathbf{Z}_{i,t}^{\text{low}}$, while highlighting the high-frequency component $\mathbf{Z}_{i,t}^{\text{high}}$. To reflect the characteristics of the representations output from TN, we take another D into consideration, as shown in Fig. 4.10a. Noticeably, the magnitude of the demand over region D is substantially smaller than those over regions A or B. Here, we account for three different times in a day, consisting of 1am, 8am and 12pm. Likewise, the input representations are plotted in Fig. 4.10b, and the intermediate representations in Fig. 4.10c. As shown in Fig. 4.10b, instances belonging to region D are mixed up without separation between different times, which signifies that the model will struggle to differentiate the times those instances occurred. By contrast, TN mitigates this issue as it forms clusters of the instances with the same occurrence time.

Chapter 5

An Adaptive, Scalable and Partially Interpretable Framework

In the dynamic landscape of time series data analysis, the quest for models that embody adaptability, scalability, and interpretability is paramount. The burgeoning complexity of data streams, characterized by intricate shifts in distribution and autocorrelation, necessitates the development of advanced forecasting frameworks. Our study embarks on this path, unveiling the Structured Component-based Neural Network (SCNN), a transformative solution that marks a new paradigm in MTS forecasting.

Our journey into the intricate world of time series data has unveiled complex shifts in distribution. **A novel revelation from our exploration is the dynamic nature of not only the distribution but also the autocorrelation within the data, exhibiting shifts both inter-days and intra-days.** This dynamic behavior underscores the need for a model capable of adeptly adapting to these shifts, thereby minimizing the necessity for manual intervention. Delving deeper into the patterns of distribution and autocorrelation, we discerned a fascinating correlation: shifts in distribution are intrinsically linked with shifts in autocorrelation

across various data subsets. This discovery hints at the potential of distribution features serving as reliable indicators for changes in autocorrelation, paving the way for more nuanced forecasting approaches. Inspired by these findings, we have come to appreciate the multifaceted roles that statistics can assume in the forecasting process. While previous studies have leveraged statistics, namely mean and standard deviation, to delineate data distribution over subsets, we now envisage these statistical components actively engaging in parameter estimation, thereby enhancing forecasting accuracy. This is primarily due to the significant influence of autocorrelation, which is correlated with data distribution, on the parameters of the forecasting model.

The SCNN stands as a beacon of innovation in this domain, adeptly decoupling the structured components of MTS data, thereby fostering enhanced interpretability and traceability. This unique approach not only facilitates deeper insights into data dynamics but also adeptly adapts to various types of space-time-varying correlations, a distinctive feature of our model. Moreover, the SCNN promises scalability, a vital attribute in handling the escalating volume and velocity of data encountered in real-world applications. It maintains a linear computational cost, avoiding the parameter increase typically associated with extended input sequence lengths. Furthermore, we acknowledge the critical importance of interpretability, a quality that grants humans a clear understanding of the decisions rendered by a model. Despite the inherent complexity of deep neural networks, which often operate as "black boxes", our endeavor is to enhance transparency, fostering trust and facilitating model debugging. This commitment to interpretability also serves to mitigate the propagation of biases present in the training data, ensuring a fair and insightful analysis.

In essence, the SCNN emerges as a robust, efficient, and insightful tool, adeptly navigating the challenges of MTS forecasting. Through its adaptive, scalable, and interpretable framework, it promises to revolutionize time series analysis, setting a new benchmark in the field.

5.1 Introduction

Multivariate time series (MTS) forecasting is a fundamental problem in the machine learning field [**?** 117]. In the era of big data, a wide array of promising applications can be conceptualized as MTS forecasting problems. Examples include predicting activities and events [42], nowcasting precipitation [50], forecasting traffic [117], and estimating pedestrian and vehicle trajectories [52]. The primary challenge in MTS forecasting is to effectively capture spatial-temporal patterns from MTS data. Spatial characteristics arise from external factors such as regional population, functionality, and geographical location. Temporal characteristics are influenced by factors like the time of day, day of the week, and weather conditions.

Traditional methods assume that the time series to be modeled is stationary [83]. However, real-world multivariate time series are often non-stationary, containing diverse and heterogeneous structured patterns such as multiple-resolution continuity and seasonality. These patterns significantly complicate the dynamics of time series, leading to various forms of distribution shifts, as illustrated in Fig. 5.1a and Fig. 5.1b. These shifts occur constantly and irregularly across hours and days, influenced by long-term continuity and seasonality. Additionally, as shown in Fig. 5.1c and Fig. 5.1d, not only does the data distribution change over time, but the auto-correlation also varies. This variation in auto-correlation, which has received little attention

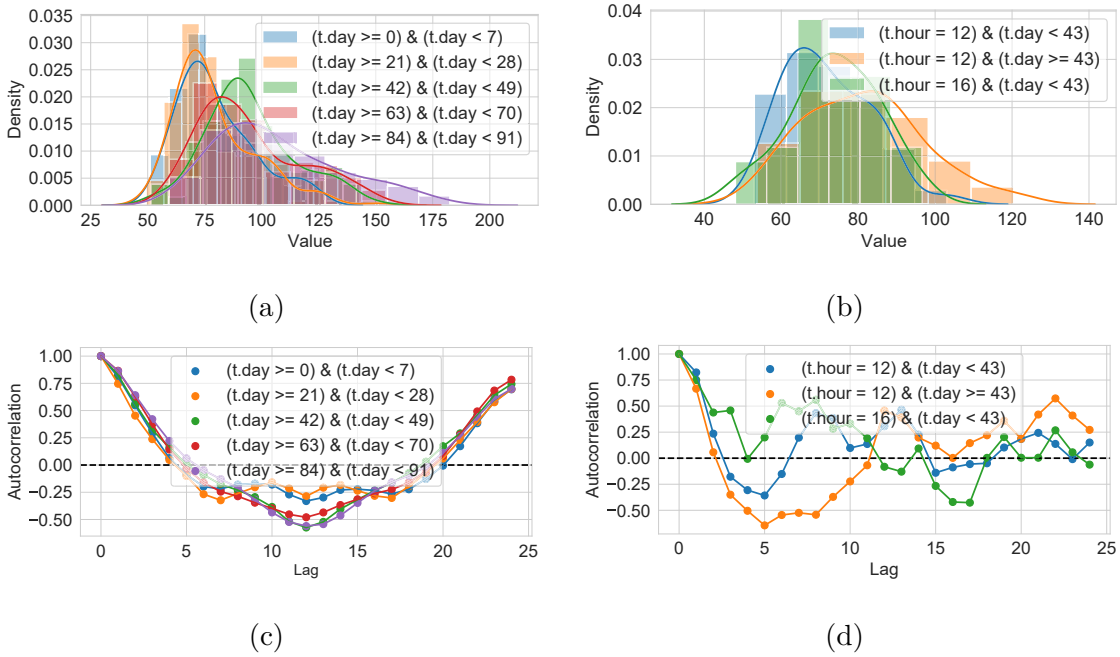


Figure. 5.1 (a) $P(Y_t|t.\text{day})$; (b) $P(Y_t|t.\text{day}, t.\text{hour})$; (c) $Corr(Y_t, Y_{t-i}|t.\text{day})$; (d) $Corr(Y_t, Y_{t-i}|t.\text{day}, t.\text{hour})$. These visualizations emphasize that both data distribution and auto-correlation exhibit complex, heterogeneous shifts correlated with factors like time span and hour of the day.

in literature, suggests that the relationships between historical observations and future targets are also unstable, making prediction more challenging.

To address non-stationary time series, modern methods employ deep neural networks like Transformers, temporal convolution networks (TCNs), and recurrent neural networks (RNNs), which do not rely on the assumption of stationarity. However, their effectiveness is limited to handling in-distribution (ID) non-stationary patterns. For example, with sine and cosine functions, their non-stationary patterns recur over time, allowing their dynamics to be captured accurately by deep learning models. However, for out-of-distribution (OOD) non-stationary patterns, the performance of these models often degrades significantly. Thus, adaptability and

generalization under complex distribution shifts remain under-explored in current deep spatial-temporal models [49, 2, 104, 103, 121]. Additionally, these methods render the prediction process a black box, lacking interpretability. They also require extensive parameters and operations, leading to prohibitively expensive computations.

Time series decomposition [83], which separates time series into trend, seasonal, and residual components, has recently emerged as a promising approach to enhance adaptability to OOD non-stationary patterns and improve interpretability of deep learning models [102, 97, 18, 63, 98]. Even simple linear models [116] have shown the ability to outperform various deep learning models [102, 129, 130] when using this approach.

Despite these advancements, current studies still have limitations. First, they focus mainly on long-term and seasonal components, capturing only coarse-grained trends while neglecting short-term or volatile components crucial for detailed deviations. Second, the segregated processing of different components without information exchange inhibits the extraction of high-order and non-linear interactions among them. Third, they overlook the dynamic nature of auto-correlation, implying the sub-optimality of using static models with fixed parameters for OOD forecasting, given that optimal parameter solutions should be governed by auto-correlation evaluations. Due to these shortcomings, previous decomposition-driven methods still rely on large-scale MLPs or Transformers to enhance model expressivity, reducing scalability and interpretability [67, 63, 123].

Our study introduces a structured component-based neural network (SCNN) for MTS forecasting. SCNN employs a divide-and-conquer strategy, strategically disentangling time series data into multiple structured components, as shown in

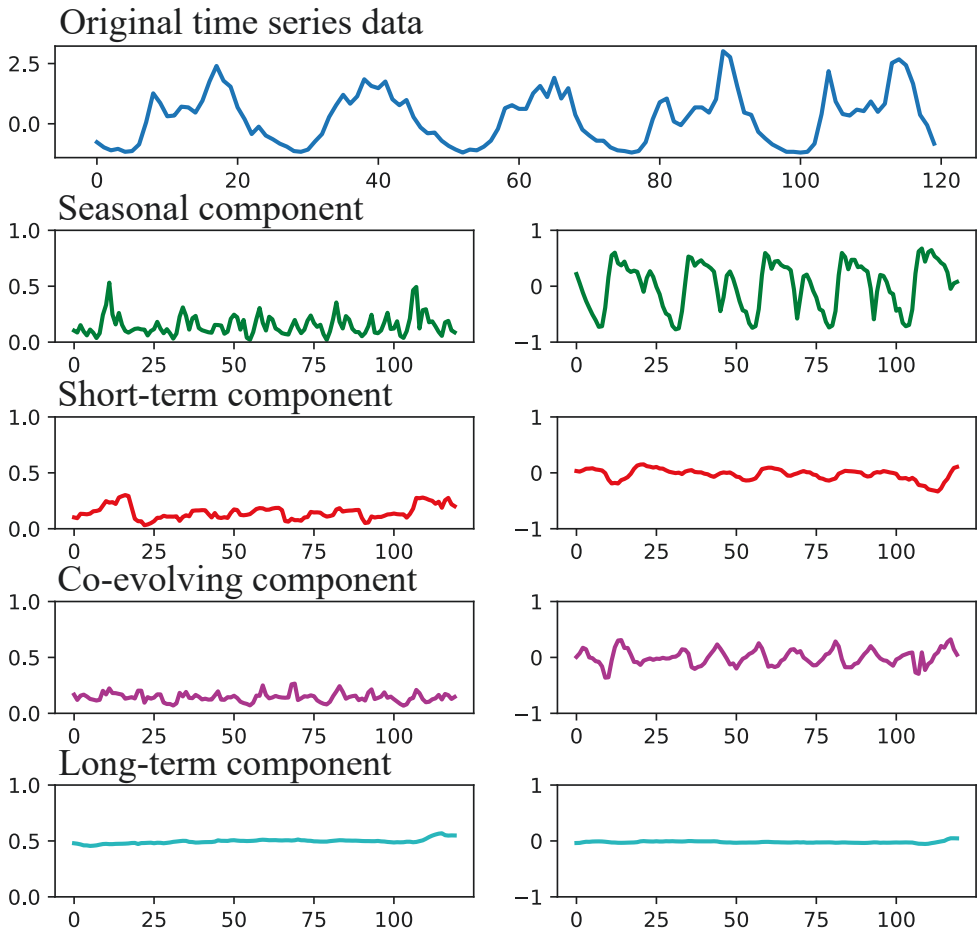


Figure. 5.2 Structured components extracted by SCNN from BikeNYC time series data. The underlying structure of TS might be far more complicated than just trend (long-term) and seasonal components.

Fig. 5.2, extending beyond long-term and seasonal components. These components exhibit heterogeneous dynamics, suitable for simulation with simple, specially-designed models. This approach significantly enhances the model's ability to handle heterogeneous distribution shifts while improving the transparency of its internal mechanisms. Unlike previous methods, where decomposition and recomposition are applied only at the input and output stages, respectively, we integrate these operations into the design of the neural modules comprising SCNN. Deep and iterative decoupling of components allows for incorporating a wide range of high-order interactions among them, thereby enhancing the model's expressivity. To address auto-correlation shifts, each neural module features a bifurcated structure, enabling dynamic and adaptive model parameter updates: one branch adjusts model parameters based on real-time data, akin to a small hyper-network [**?**], while the other processes hidden features with the adjusted parameters. Furthermore, to improve SCNN's generalization ability, we introduce auxiliary structural regularization alongside the standard regression loss. This encourages the model to focus more on structured components less prone to corruption. The components utilized in SCNN enable an adaptive, interpretable, scalable, yet powerful neural architecture for time series forecasting.

We summarize our contributions as follows:

- We introduce the Structured Component Neural Network (SCNN) for multivariate time series forecasting, marking the first completely decomposition-based neural architecture.
- We propose a novel structural regularization method to explicitly shape the structure of the representation space learned from SCNN.

- We conduct extensive experiments on three public datasets to validate the effectiveness of SCNN, and observe general improvement over competing methods.

- Empirical and analytical evidence demonstrates the SCNN's superior performance in handling distribution shifts and anomalies, while maintaining computational efficiency.

5.2 Preliminaries

In this section, we introduce the definitions and the assumption. All frequently used notations are reported in Table 5.1.

Definition 6 (Multivariate time series forecasting)**.** Multivariate time series is formally defined as a collection of random variables $\{Y_{n,t}\}_{n\in N,t\in T}$, where n denotes the index on the spatial domain and t denotes the index on the temporal domain. Time series forecasting is formulated as the following conditional distribution:

$$P(Y_{:,t+1:t+T_{\text{out}}}|Y_{:,t-T_{\text{in}}+1:t}) = \prod_{i=1}^{T_{\text{out}}} P(Y_{:,t+i}|Y_{:,t-T_{\text{in}}+1:t}).$$

Our study delves into a specific category of time series that can be represented as a superposition of various elementary signals. These include the long-term (lt) component, the seasonal (se) component, the short-term (st) component, the co-evolving (ce) component, and the residual component. Each component offers a distinct perspective on the underlying dynamic system of the time series, enriching the information content of the series.

Table 5.1 Notations

Notation	Description
N, L	Number of variables / network layers.
$T_{\text{in}}, T_{\text{out}}$	Number of input steps / output steps.
$Y \in \mathbb{R}^{N \times T}$	Multivariate time series.
$Y_{n,t}^{\text{in}} \in \mathbb{R}$	Observation of n^{th} variable at time t.
$\hat{Y}_{n,t+i}^{\text{out}} \in \mathbb{R}$	Mean prediction of the n^{th} variable for the i^{th} forecast horizon at time t .
$\hat{\sigma}_{n,t+i}^{\text{out}} \in \mathbb{R}$	Standard deviation prediction of the n^{th} variable for the i^{th} forecast horizon. at time t.
lt, se, st, ce	Abbreviations for 4 types of structured components: long-term, seasonal, short-term, co-evolving.
$\mu_{n,t}^*, \sigma_{n,t}^* \in \mathbb{R}^{d_z}$	Historical structured component.
$\hat{\mu}_{n,t+i}^*, \hat{\sigma}_{n,t+i}^* \in \mathbb{R}^{d_z}$	Extrapolation of the structured component.
$H_{n,t} \in \mathbb{R}^{8d_z}$	Concatenation of historical structured components of 4 types.
$\hat{H}_{n,t+i} \in \mathbb{R}^{8d_z}$	Concatenation of extrapolated structured components of 4 types.
$Z_{n,t}^{(l)} \in \mathbb{R}^{d_z}$	Historical residual representation at the l^{th} layer in the decoupling block.
$\hat{Z}_{n,t+i}^{(l)} \in \mathbb{R}^{d_z}$	Extrapolation of the residual representation at the l^{th} layer.
$Z_{n,t} \in \mathbb{R}^{4d_z}$	Concatenation of historical residual representations at 4 layers.
$\hat{Z}_{n,t+i} \in \mathbb{R}^{4d_z}$	Concatenation of extrapolated residual representations at 4 layers.
$S_{n,t} \in \mathbb{R}^{d_z}$	Historical state.
$\hat{S}_{n,t+i} \in \mathbb{R}^{d_z}$	Extrapolation of the state.

Definition 7 (Generative Process for Multivariate Time Series)**.** We postulate that the time series is generated through the following process:

$$Z_{n,t}^{(3)} = \sigma_{n,t}^{\text{ce}} R_{n,t} + \mu_{n,t}^{\text{ce}}, \tag{5.1}$$

$$Z_{n,t}^{(2)} = \sigma_{n,t}^{\text{st}} Z_{n,t}^{(3)} + \mu_{n,t}^{\text{st}}, \tag{5.2}$$

$$Z_{n,t}^{(1)} = \sigma_{n,t}^{\text{se}} Z_{n,t}^{(2)} + \mu_{n,t}^{\text{se}}, \tag{5.3}$$

$$Z_{n,t}^{(0)} = \sigma_{n,t}^{\text{lt}} Z_{n,t}^{(1)} + \mu_{n,t}^{\text{lt}}, \tag{5.4}$$

where $R_{n,t}$ denotes the residual component; $Z_{n,t}^{(0)}$ represents the original data, and $Z_{n,t}^{(i)}$ ($i \in \{1, 2, 3\}$) signifies the intermediate representation at the i^{th} level. Each structured component is defined by a multiplicative (scaling) factor σ_t^* and an additive factor μ_t^*, with $* \in \{\text{ce}, \text{st}, \text{se}, \text{lt}\}$.

To illustrate this generative process intuitively, consider the analysis of traffic density data. In this scenario, different components capture distinct aspects of traffic dynamics. The long-term component reflects overarching trends in traffic patterns, such as increases due to urban development or population growth. The seasonal component represents cyclical changes, like the rush hour peaks or reduced flow during off-peak times. The short-term component captures immediate, transient effects caused by events like road work or weather changes. The co-evolving component quantifies the simultaneous impact of sudden events on multiple traffic series, such as a traffic accident affecting adjacent roads. Finally, the residual component accounts for random effects, including unpredictable elements like sensor errors.

It's crucial to understand that these classifications in traffic data analysis are dynamic. For example, a sudden traffic increase at a junction might initially be considered an anomaly (residual component) but could evolve into a short-term

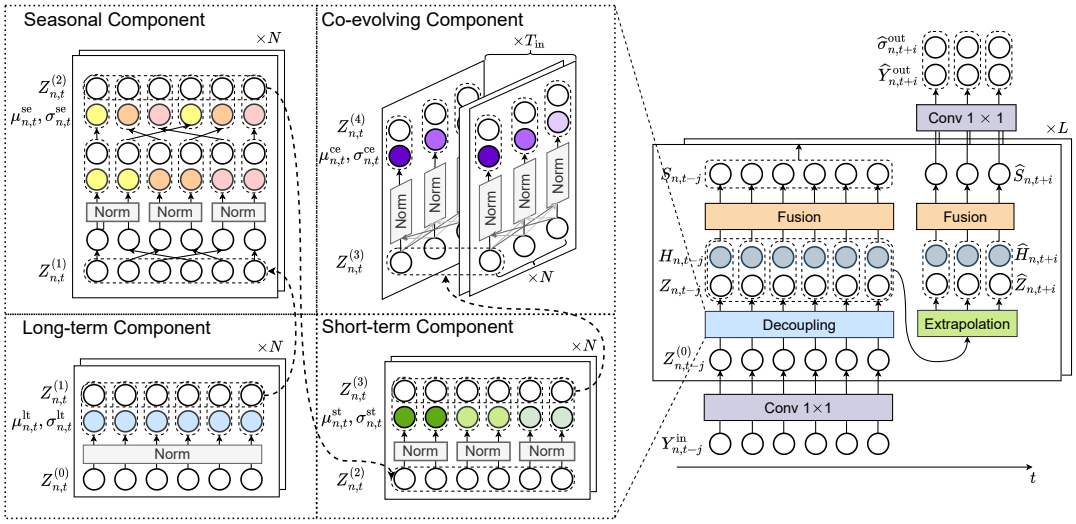


Figure. 5.3 A schematic diagram of SCNN.

pattern if it persists due to a temporary detour. If this change becomes permanent, it would then shift to the long-term component. This fluidity highlights the need for adaptable and dynamic analytical methods in traffic data analysis.

Each component in this framework exhibits both multiplicative and additive effects, reflecting the intricate nature of traffic dynamics. The multiplicative effect is vital for understanding proportional changes in traffic volume, such as varying impacts of percentage increases during peak or off-peak hours. The additive effect, on the other hand, represents uniform changes, such as the consistent impact of road constructions or new traffic signals, irrespective of current traffic levels. Incorporating both effects into each component ensures a thorough understanding of traffic dynamics, as different scenarios may necessitate focusing on either proportional (multiplicative) or absolute (additive) changes.

5.3 Structured Component-based Neural Network

Figure 5.3 illustrates an overview of our model architecture. SCNN is composed of three major parts, namely component decoupling, component extrapolation, and structural regularization. We will introduce each part in the following sections.

5.3.1 Component Decoupling

This section introduces how to estimate a specific structured component, and decouple this component from the residuals by applying a normalization operator. This process is presented in the left part of Fig. 5.3.

5.3.1.1 Long-Term Component

The long-term component aims to be the characterization of the long-term patterns of the time series data. To avoid ambiguity, we refer to the pattern as the distribution of the aggregated samples without considering the chronological order among them; the long-term pattern refers to the data distribution over an extended period that should cover multiple seasons. By aggregating the samples collected from multiple seasons, we can eliminate the short-term impact that will affect only a handful of time steps, and acquire the estimation of the long-term component with less bias.

We create a sliding window of size Δ to dynamically select the set of samples over time. Then, the location (mean) and scale (standard deviation) of the samples are computed and jointly taken as the measurement of the long-term component. Finally, we transform the representation by subtracting the location from it and dividing the difference by the scale, in order to unify the long-term components for

different samples. The formula takes the following form:

$$\mu_{n,t}^{\text{lt}} = \frac{1}{\Delta} \sum_{i=0}^{\Delta-1} Z_{n,t-i}^{(0)} \ , \tag{5.5}$$

$$(\sigma_{n,t}^{\text{lt}})^2 = \frac{1}{\Delta} \sum_{i=0}^{\Delta-1} (Z_{n,t-i}^{(0)})^2 - (\mu_{n,t}^{\text{lt}})^2 + \epsilon, \tag{5.6}$$

$$Z_{n,t}^{(1)} = \frac{Z_{n,t}^{(0)} - \mu_{n,t}^{\text{lt}}}{\sigma_{n,t}^{\text{lt}}}, \tag{5.7}$$

where $\mu_{n,t}^{\text{lt}}$ and $\sigma_{n,t}^{\text{lt}}$ are the location and the scale respectively; $Z_{n,t}^{(1)}$ is the first-layer normalized representation and passed to the following normalization layers.

Previous studies [44, 63, 18] let the ϵ to be an infinitesimal value, e.g. 0.00001, for the purpose of avoiding the division-by-zero issue. We find that this trick, however, incurs an unstable optimization process in some cases, resulting in a sub-optimal solution on the parameter space. Imagine a time series that rarely receives non-zero measurements which can be viewed as unpredictable noises. The standard deviation of this time series would be very small, leading its inverse to be exceptionally large. As a result, the noises would be undesirably magnified, driving the model to fit these chaotic patterns without any predictable structure. To alleviate this dilemma, our study sets ϵ as 1, which, on the one hand, can prevent the explosion of noises and, on the other hand, cannot dominate the original scaling factor. This simple trick is also employed by [78], but they only used it to preprocess the time series data.

When deployed for online inferencing, this process can be optimized in an iterative manner to minimize redundant computations. This optimization leverages estimators from the previous time step, updating them based on the current

observation. This update is weighted by an updating rate, denoted as α^{lt}:

$$\mu^{\text{lt}}_{n,t} = (1-\alpha^{\text{lt}})\mu^{\text{lt}}_{n,t-1} + \alpha^{\text{lt}} Z^{(0)}_{n,t} \ , \tag{5.8}$$

$$M^{\text{lt}}_{n,t} = (1-\alpha^{\text{lt}})M^{\text{lt}}_{n,t-1} + \alpha^{\text{lt}} (Z^{(0)}_{n,t})^2 \ , \tag{5.9}$$

$$(\sigma^{\text{lt}}_{n,t})^2 = M^{\text{lt}}_{n,t} - (\mu^{\text{lt}}_{n,t})^2 + \epsilon. \tag{5.10}$$

Considering that the long-term component is designed to capture sustained trends, the value of α^{lt} should be set quite low. This ensures that historical observations are retained and not dismissed quickly. It's important to note that this configuration doesn't apply to the short-term component, which is tailored to concentrate on a more limited span of observations.

5.3.1.2 Seasonal Component

Our study makes a mild assumption that the cycle length is invariant over time. For those applications with time-varying cycle lengths, we can resort to the Fast Fourier Transform (FFT) to automate the identification of cycle length, which is compatible with our framework and is applied in a bunch of methods like Autoformer [102].

Disentanglement of the seasonal component resembles the long-term component, except that we apply a dilated window whose dilation factor is set to the cycle length. Let τ denote the window size, and m denote the dilation factor. The normalization then proceeds as follows:

$$\mu^{\text{se}}_{n,t} = \frac{1}{\tau}\sum_{i=0}^{\tau-1} Z^{(1)}_{n,t-i*m} \ , \tag{5.11}$$

$$(\sigma^{\text{se}}_{n,t})^2 = \frac{1}{\tau}\sum_{i=0}^{\tau-1} (Z^{(1)}_{n,t-i*m})^2 - (\mu^{\text{se}}_{n,t})^2 + \epsilon, \tag{5.12}$$

$$Z^{(2)}_{n,t} = \frac{Z^{(1)}_{n,t} - \mu^{\text{se}}_{n,t}}{\sigma^{\text{se}}_{n,t}}. \tag{5.13}$$

In this context, the derived values of $\mu^{\text{se}}_{n,t}$ and $\sigma^{\text{se}}_{n,t}$ will solely reflect seasonal patterns, devoid of any influence from transient or short-term effects. The online inferencing procedure is then executed as outlined below, governed by a distinct updating rate α^{se}:

$$\mu^{\text{se}}_{n,t} = (1-\alpha^{\text{se}})\mu^{\text{se}}_{n,t-m} + \alpha^{\text{se}} Z^{(1)}_{n,t} \ , \tag{5.14}$$

$$M^{\text{se}}_{n,t} = (1-\alpha^{\text{se}})M^{\text{se}}_{n,t-m} + \alpha^{\text{se}} (Z^{(1)}_{n,t})^2 \ , \tag{5.15}$$

$$(\sigma^{\text{se}}_{n,t})^2 = M^{\text{se}}_{n,t} - (\mu^{\text{se}}_{n,t})^2 + \epsilon. \tag{5.16}$$

The value of α^{se} depends on how long memory of the seasonal component the practitioner wants to keep.

5.3.1.3 Short-Term Component

The short-term component captures the irregular and short-term effects, which cannot be explained by either the long-term component or the seasonal component. In contrast to the long-term normalization, the window size here needs to be set to a small number, notated by δ, such that the short-term effect will not be smoothed out. Likewise, the formula takes the following form:

$$\mu^{\text{st}}_{n,t} = \frac{1}{\delta}\sum_{i=0}^{\delta-1} Z^{(2)}_{n,t-i} \ , \tag{5.17}$$

$$(\sigma^{\text{st}}_{n,t})^2 = \frac{1}{\delta}\sum_{i=0}^{\delta-1} (Z^{(2)}_{n,t-i})^2 - (\mu^{\text{st}}_{n,t})^2 + \epsilon, \tag{5.18}$$

$$Z^{(3)}_{n,t} = \frac{Z^{(2)}_{n,t} - \mu^{\text{st}}_{n,t}}{\sigma^{\text{st}}_{n,t}}. \tag{5.19}$$

Unlike the long-term component, the online inferencing procedure for the short-term component utilizes the same expression but is regulated by a higher updating rate

α^{st}:

$$\mu_{n,t}^{\text{st}} = (1-\alpha^{\text{st}})\mu_{n,t-1}^{\text{st}} + \alpha^{\text{st}} Z_{n,t}^{(2)} \ , \tag{5.20}$$

$$M_{n,t}^{\text{st}} = (1-\alpha^{\text{st}})M_{n,t-1}^{\text{st}} + \alpha^{\text{st}} (Z_{n,t}^{(2)})^2 \ , \tag{5.21}$$

$$(\sigma_{n,t}^{\text{st}})^2 = M_{n,t}^{\text{st}} - (\mu_{n,t}^{\text{st}})^2 + \epsilon. \tag{5.22}$$

The downside of the short-term component is that it cannot timely detect a short-term change in data, demonstrating response latency. Also, it is insensitive to changes that only endure for a limited number (e.g., two or three) of time steps. To mitigate this issue, we can make use of the contemporary measurements of the co-evolving time series.

5.3.1.4 Co-evolving Component

The co-evolving component, derived from the spatial correlations between time series, is advantageous for capturing instant changes in time series, which distinguishes it from the above three components. A co-evolving behavior shared across multiple time series indicates that these time series are generated from the same process. Then, we can get an estimator of this process by aggregating multiple samples drawn from it.

A key problem to be solved here is identifying which time series share the same co-evolving component. Technically, this amounts to measuring correlations between different time series. This measurement can be done either by hard-coding the correlation matrix with prior knowledge or by parameterizing and learning it. Our study adopts the latter practice, which allows for more flexibility, since many datasets do not present prior knowledge about the relationship between time series. We assign an individual attention score to every pair of time series, and then

normalize the attention scores associated with the same time series via softmax to ensure that all attention scores are summed up to 1. Formally, let $\alpha_{n,n'}$ denote the attention score between the n^{th} and n'^{th} variable. The formula is written as follows:

$$a_{n,n'} = \frac{\exp(\alpha_{n,n'})}{\sum_{j=1}^{N} \exp(\alpha_{n,j})}, \tag{5.23}$$

$$\mu_{n,t}^{\text{ce}} = \sum_{n'=1}^{N} a_{n,n'} Z_{n',t}^{(3)} , \tag{5.24}$$

$$(\sigma_{n,t}^{\text{ce}})^2 = \sum_{n'=1}^{N} a_{n,n'} (Z_{n',t}^{(3)})^2 - (\mu_{n,t}^{\text{ce}})^2 + \epsilon, \tag{5.25}$$

$$Z_{n,t}^{(4)} = \frac{Z_{n,t}^{(3)} - \mu_{n,t}^{\text{ce}}}{\sigma_{n,t}^{\text{ce}}}, \tag{5.26}$$

where $Z_{n,t}^{(4)}$ denotes the residuals that cannot be modeled by any of our proposed components. This computation can be further modified to improve the scalability via the adjacency matrix learning module proposed in [104].

The decoupled components and residual representations are sequentially concatenated to form a wide vector:

$$\begin{aligned} Z_{n,t} =& [Z_{n,t}^{(1)} , Z_{n,t}^{(2)} , Z_{n,t}^{(3)} , Z_{n,t}^{(4)}], \\ H_{n,t} =& [\mu_{n,t}^{\text{lt}} , \sigma_{n,t}^{\text{lt}} , \mu_{n,t}^{\text{se}} , \sigma_{n,t}^{\text{se}} , \\ & \mu_{n,t}^{\text{st}} , \sigma_{n,t}^{\text{st}} , \mu_{n,t}^{\text{ce}} , \sigma_{n,t}^{\text{ce}}]. \end{aligned}$$

5.3.2 Component Extrapolation

We simulate the dynamics of each component with a customized and basic model. This allows for the explainability of the features being accounted for by the model and the provision of insights into the capacity of the forecasting model. With the acquired understanding of the features and the model capacity, practitioners can detect the anomaly points where the model may not present reliable results, and

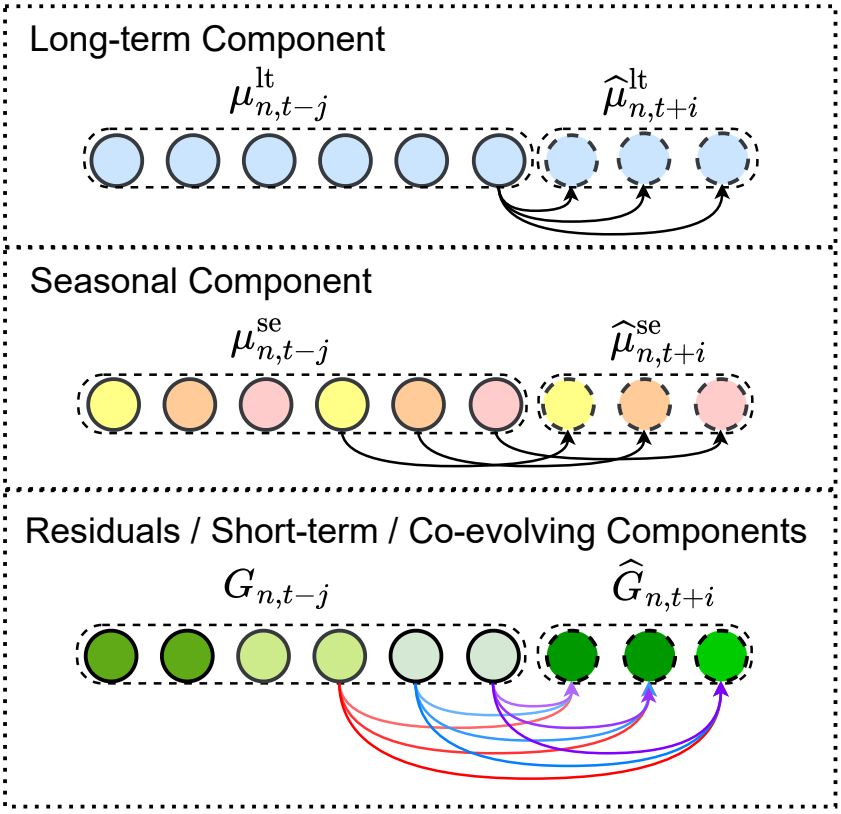


Figure. 5.4 Component Extrapolation

adopt specific measures to handle the anomalies. The components exhibit different dynamics with varying degrees of predictability, motivating us to create separate models to mimic the prospective development of their dynamics. The models are visualized in Fig. 5.4.

For a short period of time in the future, the long-term component and the seasonal component change in a relatively well-defined behavior, so we can directly specify the law for extrapolation without introducing extra parameters. We trivially reuse the (estimated) state of the long-term component at the current time point for the extrapolation of each future time point.

$$\hat{\mu}^{\text{lt}}_{n,t+i} = \mu^{\text{lt}}_{n,t} \ , \ \hat{\sigma}^{\text{lt}}_{n,t+i} = \sigma^{\text{lt}}_{n,t} \ . \tag{5.27}$$

For the seasonal component and its residual, we also conduct replication but from the time point at the same phase as the target time point in the previous season:

$$\hat{\mu}^{\text{se}}_{n,t+i} = \mu^{\text{se}}_{n,t-m+i} \ , \ \hat{\sigma}^{\text{se}}_{n,t+i} = \sigma^{\text{se}}_{n,t-m+i} \ . \tag{5.28}$$

The short-term component, the co-evolving component, and the residual representations vary with greater stochasticity and thereby less predictability than the above two components due to their irregularity. Since the dynamics are now much more complicated, we opt to parameterize the dynamical model to allow for more flexibility than specifying a fixed heuristic law. For each of these three types of representations, we employ an auto-regressive model, predicting the representation for the i^{th} forecast horizon based on the past δ representations. For the sake of brevity, we present the extrapolation processes of the short-term and co-evolving components together with the residuals in a single figure, given that they share the same model form:

$$\hat{G}_{n,t+i} = \sum_{j=0}^{\delta-1} \hat{W}_{ji} G_{n,t-j} + b_i, \tag{5.29}$$

where $G \in \{Z^{(l)}_{n,t+i}, \mu^{\text{st}}_{n,t+i}, \sigma^{\text{st}}_{n,t+i}, \mu^{\text{ce}}_{n,t+i}, \sigma^{\text{ce}}_{n,t+i}\}$; $\hat{W}_{ji}$, a parameter matrix of size $d_z \times d_z$, quantifies the contribution from $G_{n,t-j}$ to $\hat{G}_{n,t+i}$; b_i is the bias term. $\hat{W}_{ji}$ and b_i are subject to training.

We concatenate the extrapolated components, denoted as $\hat{H}_{n,t+i}$, and the residuals, $\hat{Z}_{n,t+i}$. We then model their interactions, parameterized by two learnable matrices, $\hat{W}^{(1)}$ and $\hat{W}^{(2)}$, both belonging to $\mathbb{R}^{d_z \times 12 d_z}$, as follows:

$$\begin{aligned} \hat{S}_{n,t+i} = &\left(\hat{W}^{(1)}[\hat{Z}_{n,t+i}, \hat{H}_{n,t+i}]\right) \\ &\otimes \left(\hat{W}^{(2)}[\hat{Z}_{n,t+i}, \hat{H}_{n,t+i}]\right), \end{aligned} \tag{5.30}$$

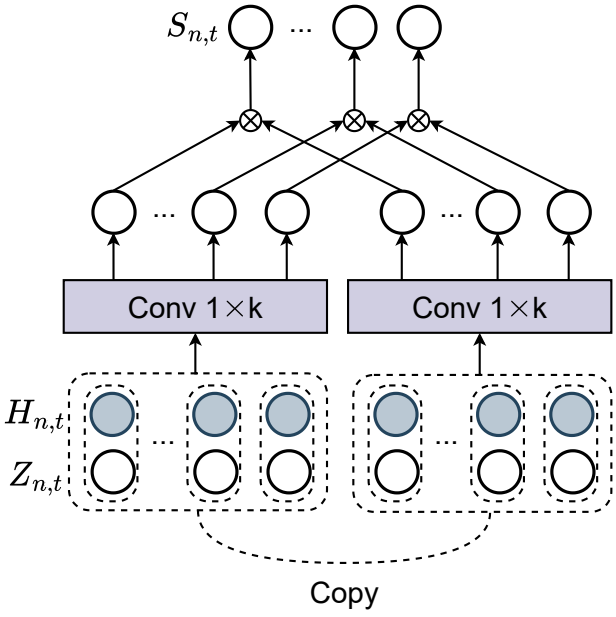


Figure. 5.5 Component Fusion

So far, we construct a projection from the past to the future, consisting of statistically meaningful operations.

5.3.3 Component Adaptive Fusion

As illustrated in Fig. 5.1a, there is a notable divergence in both the data distribution and the temporal correlation, observed both intra-days and inter-days. While the temporal correlation holds significance comparable to the data distribution, it has been relatively overlooked in research and discussions. At its core, the model aims to discern the temporal correlations between forward and backward observations. Consequently, these correlations are intrinsically embedded within the model parameters. Recognizing and adapting to the subtle shifts in temporal correlations can enhance forecasting accuracy.

To equip the model with the capability to discern when and how these temporal correlations evolve, structured components prove beneficial. A closer examination of Fig. 5.1a reveals a correlation between shifts in temporal correlations and shifts in data distributions. This observation implies that structured components can also serve as indicators of temporal correlations. Therefore, these components serve a dual purpose in forecasting: they capture both data distribution patterns and temporal correlations. To fully harness the capabilities of structured components, we introduce a neural module bifurcated into two branches: one dedicated to feature learning and the other to parameter learning. The outputs from these branches are then amalgamated using an element-wise multiplication operation. For the sake of simplicity, each branch employs a convolution operator, though this can be augmented with more intricate operations, such as MLP. This computational process is graphically represented in Fig. 5.5, and is formally written as:

$$S_{n,t} = \left(\sum_{j=0}^{k-1} W_j^{(1)}[Z_{n,t-j}, H_{n,t-j}]\right) \otimes \left(\sum_{j=0}^{k-1} W_j^{(2)}[Z_{n,t-j}, H_{n,t-j}]\right), \tag{5.31}$$

where k is the kernel size of the convolution operator and $W_j^{(1)}, W_j^{(2)} \in \mathbb{R}^{d_z \times 12 d_z}$ are learnable matrices. $S_{n,t}$ can be passed to another component estimation block as $Z_{n,t}^{(0)}$ to produce richer compositions of the structural components.

5.3.4 Structural Regularization

Conventionally, the objective function for time series forecasting aims to minimize the mean squared errors (MSE) or mean absolute errors (MAE) between the predictions and the ground truth observations. The assumption inherent to this

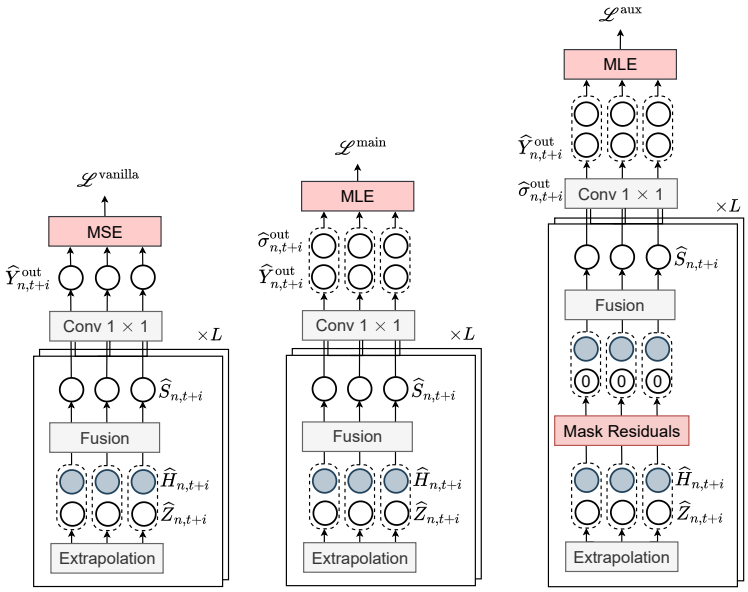


Figure. 5.6 Structural Regularization. The term $\mathcal{L}^{\text{vanilla}}$ denotes the standard MSE loss function. On the other hand, $\mathcal{L}^{\text{main}}$ and $\mathcal{L}^{\text{aux}}$ are specifically designed to enforce regularization within the feature space, thereby ensuring a more structured representation of the data. These two loss functions work together to optimize the model's performance.

objective is that all the variables share the same variance of 1. However, this does not enable the learned representations to be organized in a desired structure, where variables can see different degrees of variance at different times due to the time-varying scaling effects prescribed by the generative structure of time series. Instead, we opt to optimize the maximum likelihood estimate (MLE) [78], which allows SCNN to improve the shaping of the structure of the representation space. In addition, an auxiliary objective function is designed to improve the nuances in feature space at the component level. We graphically contrast the two designed objective functions against the vanilla MSE loss Fig. 5.6

We apply linear transformations to the representations output from the component extrapolation module, producing the location (i.e. mean) $\hat{Y}^{\text{out}}_{n,t+i}$ and the scale (i.e. standard deviation) $\hat{\sigma}^{\text{out}}_{n,t+i}$, where $\hat{\sigma}^{\text{out}}_{n,t+i}$ further goes through a SoftPlus function to enable itself to be non-negative. The MLE loss is written as:

$$\mathcal{L}^{\text{main}} = \sum_{n=1}^{N}\sum_{i=1}^{T_{\text{out}}}(\log(\text{SoftPlus}(\hat{\sigma}^{\text{out}}_{n,t+i})) + \frac{(Y_{n,t+i} - \hat{Y}^{\text{out}}_{n,t+i})^2}{2(\text{SoftPlus}(\hat{\sigma}^{\text{out}}_{n,t+i}))^2}).$$

The first term in the above loss function encourages the scaling factor to be small, and the second term penalizes the deviation between the extrapolated data and the ground truth data weighted by the inverse of the scaling factor.

Solely leveraging the above objective to learn the forecasting dynamics does not ensure robust estimation of the structured components with their contribution to the projection. The intuition is that since the residual components, especially at the bottom levels, still contain a part of the structural information, they will take a certain amount of attributions that are supposed to belong to the structured components as learning the corresponding weights for the components. Attributing improper importance to the residual components incurs considerable degradation

in the model performance, once the time series data is contaminated with random noise that heavily impacts the high-frequency signal.

To approach this issue, the basic idea is to accentuate the structured components that suffer less from corruption with an additional regularizer. This regularizer works to prompt the model to achieve a reasonable forecast using purely the structured components without the need for residual components. In particular, in the forward process of a training iteration, SCNN forks another branch after the component extrapolation module. This branch starts by zero-masking all the residual components, passing only structured components through the following operations. Finally, it yields an auxiliary pair of forecast coefficients $\hat{Y}^{\text{aux}}_{n,t+i}$ and $\hat{\sigma}^{\text{aux}}_{n,t+i}$, which are also being tailored by MLE.

The ultimate objective to be optimized is an aggregation of all the above objective functions in a weighted fashion:

$$\mathcal{L} = \alpha \mathcal{L}^{\text{aux}} + \mathcal{L}^{\text{main}}, \tag{5.32}$$

where α is the hyper-parameter that controls the importance of the corresponding objective. We use the Adam optimizer [46] to optimize this target.

5.3.5 Complexity Analysis

We conduct an analysis of two types of complexity associated with our model: first, the parameter complexity, which refers to the number of parameters involved in the model; and second, the computational complexity. We draw a comparison between the complexity of the SCNN and three prominent frameworks, namely the Transformer, the TCN, and the MLP.

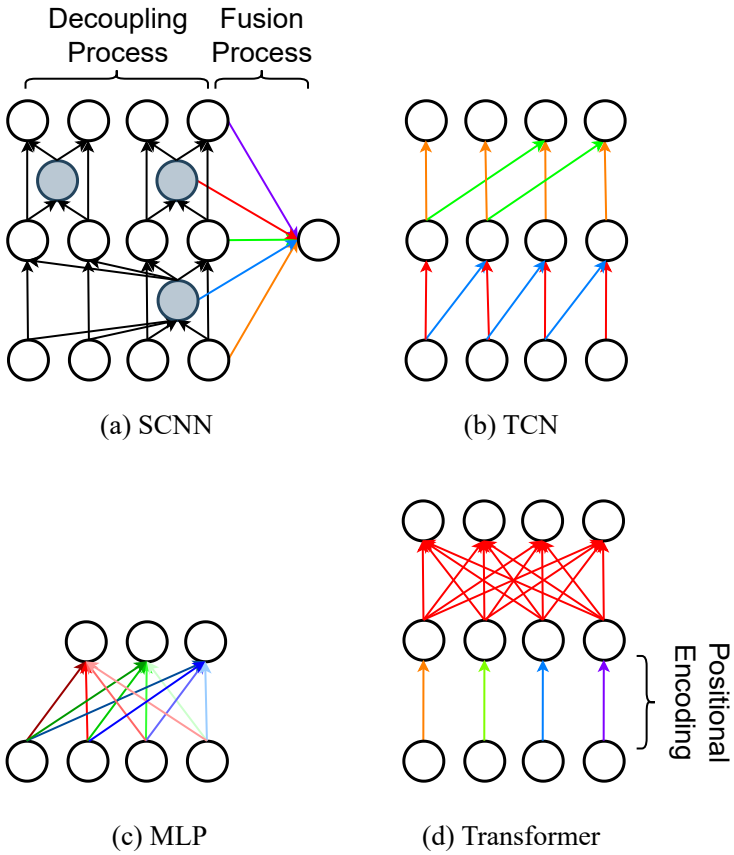


Figure. 5.7 Data and computational flow. Each edge symbolizes an atomic operation involving a single variable situated at the tail of the edge. If an operation is parameterized, the corresponding edge is color-coded.

Figure 5.7 provides a visual representation of the data and computational flow associated with these four frameworks. Within these diagrams, each edge symbolizes an atomic operation involving a single variable situated at the tail of the edge. If an operation is parameterized, the corresponding edge is color-coded. Edges sharing the same color denote operations utilizing the same set of learnable parameters. Within the SCNN framework, the decoupling process is carried out without parameterization, thus these edges are illustrated in black. The structured components that emerge from this process are subsequently integrated, employing component-dependent parameters.

Let's denote the number of components crafted within our model as m. The number of parameters within SCNN scales in proportion to the number of components inherent in the time series, which is $\mathcal{O}(m)$. This contrasts with the majority of state-of-the-art (SOTA) models, where the parameter count scales with the length of the input sequence. To illustrate, TCN or WaveNet-based models necessitate at least $\mathcal{O}(\log T)$ parameters to process a sequence of length T; MLP or Linear Regression (LR)-based models require $\mathcal{O}(T)$ parameters; and Transformer-based models also demand $\mathcal{O}(T)$ parameters to attain SOTA performance, as demonstrated in [123]. Our approach aligns with the principle that the complexity of the underlying dynamical system dictates the requisite number of parameters, regardless of the input sequence length.

Regarding the computational complexity relative to sequence length, SCNN attains a complexity of $\mathcal{O}(Tm)$. Notably, we can further reduce the complexity of an inference step to $\mathcal{O}(m)$ by approximating the structured component using a moving average approach. This stands in contrast to alternative methods such as the MLP, which achieves a complexity of $\mathcal{O}(Th)$, with h representing the number

of units in the hidden layer. The Transformer model yields a complexity of $\mathcal{O}(T^2)$, while the TCN model reaches a complexity of $\mathcal{O}(T \log T)$. Therefore, in terms of computational complexity with respect to sequence length, the SCNN proves to be the most efficient model, particularly when the structured component is estimated in a moving average manner. This observation underscores the advantage of SCNN in scenarios where computational efficiency and scalability are critical considerations.

5.4 Evaluation

In this section, we conduct extensive experiments on three common datasets to validate the effectiveness of SCNN from various aspects.

5.4.1 Experiment Setting

5.4.1.1 Datasets

To evaluate the performance of our model, we conduct experiments on three real-world datasets, namely BikeNYC[1], PeMSD7[2] and Electricity[3]. The statistics and the experiment settings regarding the three datasets are reported in Table 5.2. We adopt the same data pre-processing strategy as most of the current works [104, 103], where the TS data of each variable is individually standardized.

5.4.1.2 Network Setting

The input length is set to a multiple of the season length, so that sufficient frames governed by approximately the same seasonal and long-term components can be

[1]https://ride.citibikenyc.com/system-data
[2]https://pems.dot.ca.gov/
[3]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams 20112014

Table 5.2 Dataset statistics.

Tasks	Electricity	PeMSD7	BikeNYC
Start time	10/1/2014	5/1/2012	4/1/2014
End time	12/31/2014	6/30/2012	9/30/2014
Sample rate	1 hour	30 minutes	1 hour
# Timesteps	2184	2112	4392
# Variate	336	228	128
Training size	1848	1632	3912
Validation size	168	240	240
Testing size	168	240	240
Input length	144	288	144
Output length	3	3	3

Table 5.3 Performance on the BikeNYC dataset

Model	MAPE (%)			MAE			RMSE		
	Horizon 1	Horizon 2	Horizon 3	Horizon 1	Horizon 2	Horizon 3	Horizon 1	Horizon 2	Horizon 3
Autoformer	20.1	21.1	22.8	3.01	3.11	3.38	6.17	6.44	7.08
LSTNet	21.2	22.3	23.8	2.71	2.91	3.15	5.80	6.34	6.97
StemGNN	19.0	20.8	22.5	2.50	2.74	2.93	5.25	6.09	6.62
AGCRN	17.4	18.8	20.5	2.28	2.50	2.68	4.74	5.50	5.97
GW	18.2	19.5	20.9	2.35	2.57	2.75	4.83	5.56	6.06
MTGNN	18.0	19.5	20.9	2.35	2.57	2.73	4.87	5.69	6.18
SCINet	17.9	19.8	21.4	2.38	2.68	2.94	4.88	5.78	6.60
STG-NCDE	18.7	20.6	22.2	2.40	2.67	2.90	5.04	5.86	6.56
GTS	20.6	23.6	26.7	2.38	2.58	2.74	4.85	5.53	6.01
ST-Norm	17.3	18.6	19.9	2.26	2.46	2.62	4.66	5.38	5.84
SCNN	**16.5**	**17.3**	**18.4**	**2.13**	**2.27**	**2.40**	**4.44**	**5.02**	**5.42**
Imp	+4.6%	+6.9%	+7.5%	+5.7%	+7.7%	+8.3%	+4.7%	+6.6%	+7.1%

Table 5.4 Performance on the PeMSD7 dataset

Model	MAPE (%)			MAE			RMSE		
	Horizon 1	Horizon 2	Horizon 3	Horizon 1	Horizon 2	Horizon 3	Horizon 1	Horizon 2	Horizon 3
Autoformer	9.01	9.41	9.86	4.57	4.75	5.03	6.85	7.15	7.38
LSTNet	7.48	7.77	8.19	3.58	3.71	3.90	6.24	6.40	6.64
StemGNN	5.50	7.33	8.09	2.65	3.49	3.84	4.55	5.99	6.53
AGCRN	4.97	6.49	7.21	2.35	3.02	3.34	4.29	5.57	6.10
GW	5.02	6.56	7.10	2.39	3.10	3.35	4.28	5.51	5.94
MTGNN	5.32	6.71	7.31	2.57	3.15	3.44	4.36	5.56	6.01
SCINet	5.16	6.72	7.23	2.47	3.18	3.45	4.31	5.60	6.05
STG-NCDE	4.94	6.63	7.58	2.32	3.06	3.47	4.42	5.91	6.70
GTS	5.35	6.97	7.70	2.53	3.26	3.58	4.42	5.74	6.30
ST-Norm	4.76	6.27	7.03	2.27	2.98	3.36	4.21	5.54	6.07
SCNN	**4.47**	**5.92**	**6.50**	**2.10**	**2.75**	**2.99**	**4.06**	**5.29**	**5.76**
Imp	+6%	+5.5%	+7.5%	+7.4%	+7.7%	+10%	+3.5%	+3.9%	+3.7%

gathered to yield estimation without much deviation. The layer number is set to 4; The number of hidden channels d is 8; Δ is set to the same quantity as the length of the input sequence; δ is set to 8; the kernel size of the causal convolution k is configured as 2. In the training phase, the batch size is 8; the weight for the

Table 5.5 Performance on the Electricity dataset

Model	MAPE (%)			MAE			RMSE		
	Horizon 1	Horizon 2	Horizon 3	Horizon 1	Horizon 2	Horizon 3	Horizon 1	Horizon 2	Horizon 3
Autoformer	22.1	22.1	21.9	32.5	32.4	32.5	67.0	68.0	68.7
LSTNet	22.4	23.0	24.8	31.1	31.8	33.8	61.2	62.6	66.8
StemGNN	10.8	13.7	15.7	15.5	19.6	22.3	34.3	43.9	49.7
AGCRN	11.4	15.6	18.0	17.3	23.0	26.4	38.9	51.2	57.9
GW	11.3	15.6	17.3	16.3	22.0	24.3	32.5	43.6	48.7
MTGNN	10.2	13.9	16.0	14.4	19.4	22.2	29.8	40.3	46.5
SCINet	10.3	13.7	16.2	14.7	20.2	23.6	33.2	44.0	51.7
STG-NCDE	10.9	14.2	16.0	16.2	21.1	23.7	36.3	47.7	52.9
GTS	10.0	14.2	17.1	14.1	19.0	22.1	31.6	42.5	48.2
ST-Norm	10.2	13.2	15.3	15.2	19.8	22.8	32.3	42.9	50.2
SCNN	**7.69**	**10.5**	**12.2**	**11.1**	**15.0**	**17.3**	**23.9**	**32.9**	**38.4**
Imp	+23.1%	+20.4%	+20.2%	21.9%	+20.9%	+21.7%	+19.7%	+18.3%	+17.4%

auxiliary objective α is 0.5; the learning rate of the Adam optimizer is 0.0001. We also test other configurations in the hyper-parameter analysis.

5.4.1.3 Evaluation Metrics

We validate our model by root mean squared error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE). We repeat the experiment ten times for each model on each dataset and report the mean of the results.

5.4.2 Baseline Models

We compare SCNN with the following state-of-the-art models:

- **Autoformer[102].** Autoformer is specifically designed to incorporate a decomposition mechanism within the Transformer framework.
- **LSTNet[49].** LSTNet uses CNN to extract local features and uses RNN to capture long-term dependencies. It also employs a classical auto-regressive model to address scale-insensitive limitations.
- **StemGNN[6].** StemGNN models spatial and temporal dependencies in the spectral domain.

- **GW[104].** GW proposes an adaptive graph learning module that progressively recovers the spatial correlations during training. In addition, it employs Wavenet to handle correlations in the temporal domain.

- **MTGNN[103].** MTGNN designs a graph learning module that integrates external knowledge like variable attributes to learn uni-directed relations among variables.

- **AGCRN[2].** AGCRN develops two adaptive modules to build interactions between the variables. In addition, it selects RNN to undertake the job of modeling temporal evolution.

- **SCINet[61]** SCINet proposes a downsample-convolve-interact architecture which is beneficial for integrating multi-resolution features.

- **STG-NCDE[12].** STG-NCDE takes advantage of Neural Controlled Differential Equations (NCDEs) to conduct spatial-temporal processing. It generalizes canonical RNN and CNN to continuous RNN and GCN based on NCDEs.

- **GTS[80].** GTS proposes a structure learning module to learn pairwise relationships between the variables.

- **ST-Norm[18].** ST-Norm designs two normalization modules to refine the high-frequency and local components separately from MTS data.

In order to make the comparison fair, all the competing models are fed with the same number of preceding frames as SCNN. We find that this extension of input horizons can bring performance gain to various degrees.

5.4.3 Experiment Results

The experiment results on the three datasets are respectively reported in Table 5.3, Table 5.4, and Table 5.5. It is evident that the performance of SCNN surpasses that of the baseline models by 4% to 20%, especially when performing forecasts for multi-step ahead. This is because SCNN can extract the structured components with a well-conditioned deviation. As we know, raw data contains much noise, unavoidably interfering with the quality of the extracted components. SCNN can effectively deal with this issue according to the central limit theorem. In contrast, all the benchmark models, except ST-Norm, did not explicitly account for the structured components. For example, SCINet, one of the most up-to-date state-of-the-art models, struggled to achieve competitive performance in short-term MTS forecasting, due to its deficiency in adapting to the short-term distribution shift even with the enhancement of RevIN module proposed by [44]. GTS, GW, MTGNN and AGCRN were capable of learning the spatial correlations across the variables to estimate the translating effect of a co-evolving component, but were insusceptible to the changes in its scaling effects over time. ST-Norm could decouple the long-term component and the global component (a reduced form of co-evolving component), but did not introduce the constraint to the structure of feature space.

5.4.3.1 Adaptability

The data patterns for the first and last few days covered by the datasets are compared in Fig. 5.8. The solid line denotes the seasonal mean of MTS; the bind denotes the evolution of the interval between (mean - std, mean + std). It is worth noting that the data patterns for the three datasets, especially the Electricity dataset, show systematic changes from the beginning to the end. As

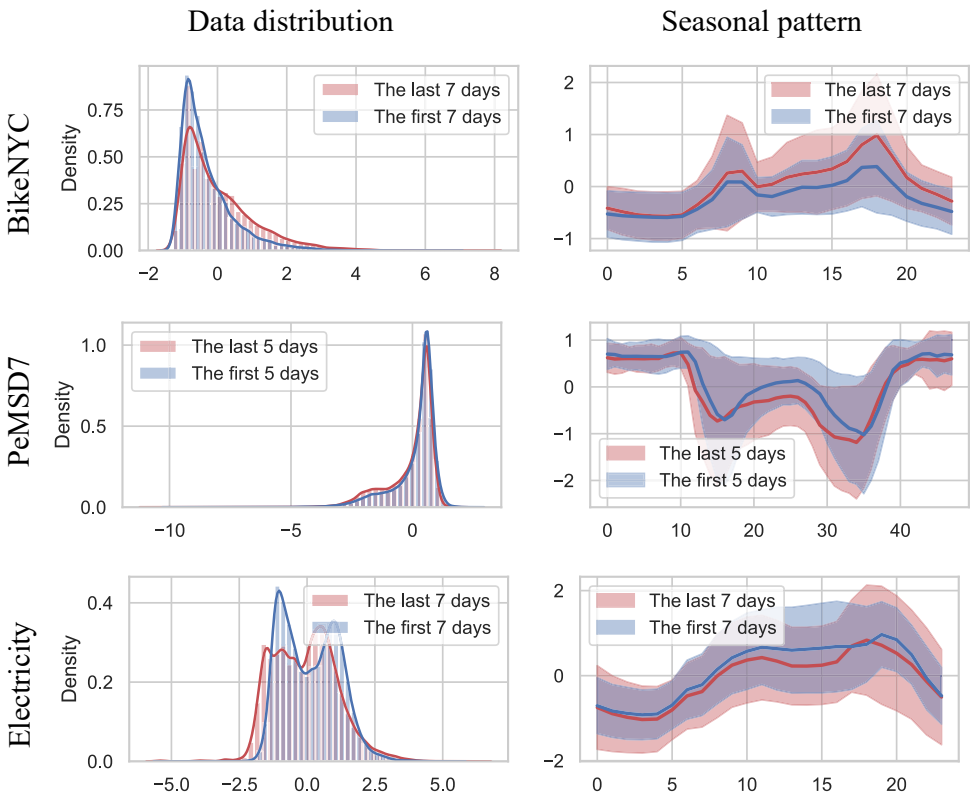


Figure. 5.8 Changes in data patterns as time evolves.

SCNN captures the data patterns on the fly, it can automatically adapt to these statistical changes, which explains that the performance of SCNN, especially when evaluated on Electricity, exceeds that of the other competing methods by a wide margin.

5.4.4 Ablation Study

We design several variants, each of which is without a specific ingredient to be validated. We evaluate these variants on all three datasets and report the overall results on RMSE in Table 5.6. It is evident that each component can contribute to the performance of the model, but to different degrees across the three datasets.

Table 5.6 Ablation Study

Models	BikeNYC	PeMSD7	Electricity
w/o μ^{lt} and σ^{lt}	5.12	<u>5.04</u>	32.7
w/o μ^{se} and σ^{se}	5.35	5.37	37.8
w/o μ^{st} and σ^{st}	5.11	5.08	33.7
w/o μ^{ce} and σ^{ce}	5.56	5.17	32.5
w/o scaling	<u>4.98</u>	5.05	35.6
w/o non-negligible ϵ	5.50	5.12	**30.6**
vanilla MSE loss	5.22	5.10	32.1
SCNN	**4.96**	**5.03**	<u>31.0</u>

The co-evolving component is ranked as the most advantageous component in the BikeNYC task. This is because the co-evolving component incorporates the spectrum of effects ranging from long-term to short-term, and can be estimated with reasonable accuracy when the number of co-evolving variables is adequately large, which is the case for the BikeNYC data. The modeling of the long-term component only brings incremental gain to the PeMSD7 task since the training data and the testing data share an identical distribution. The scaling transformation results in significant improvement in the Electricity dataset, owing to its unification of the variables showing great differences in variance. The non-negligible ϵ, as introduced in the last paragraph of Sec. 5.3.1.1, is particularly useful for training SCNN on the BikeNYC dataset, as a part of TS in this dataset is very scarce, having only a handful of irregular non-zero measurements. In contrast to the vanilla MSE loss, the structural regularization can shape the structure of the feature space, preventing the overfitting issue and unlocking more power from the structured components.

5.4.5 Hyper-Parameter Analysis

As shown in Fig. 5.9a, it is surprising that a 2-layer SCNN achieves fairly good performance, and more layers only result in incremental improvements. This

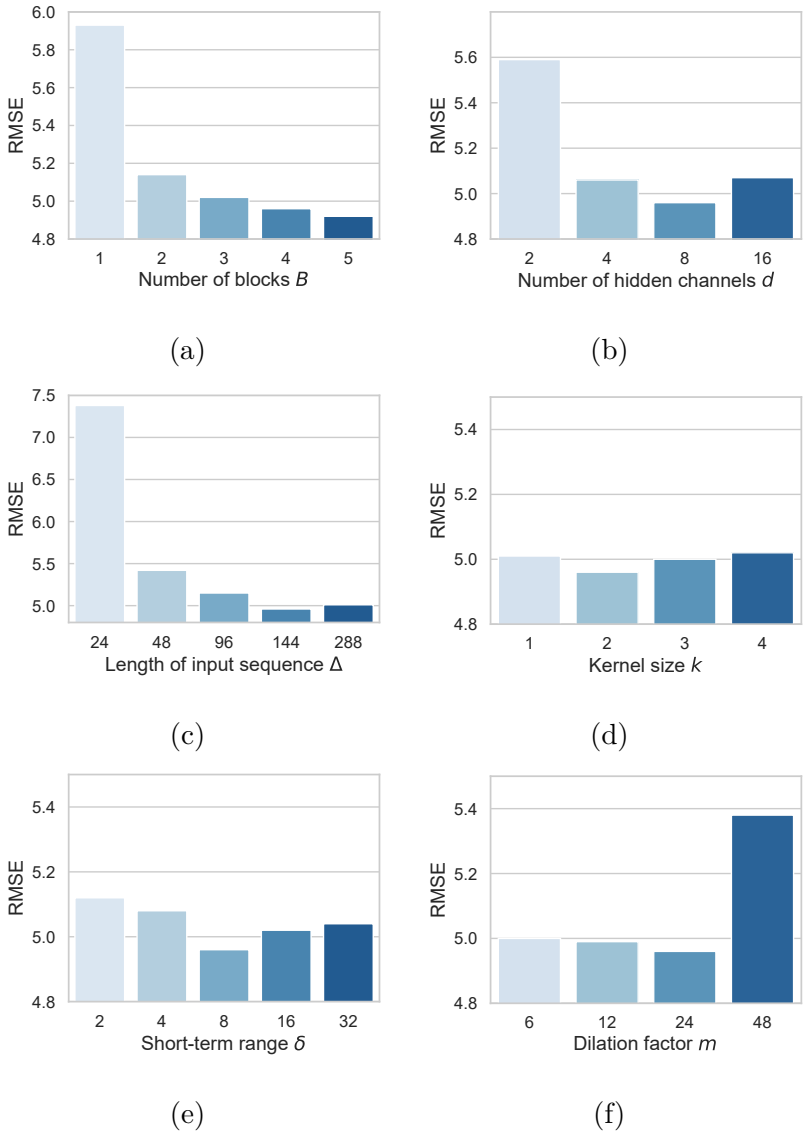


Figure. 5.9 Hyper-parameter analysis on BikeNYC data.

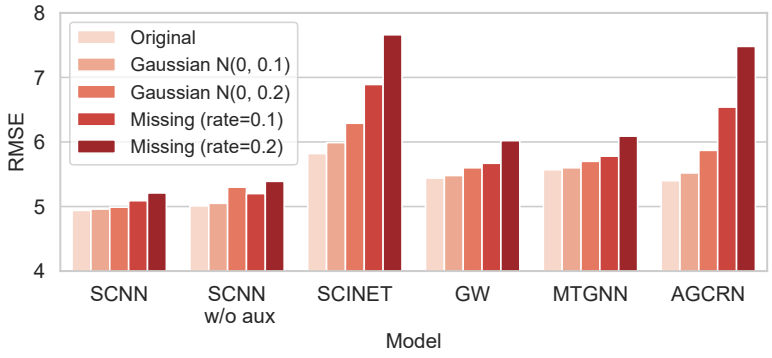
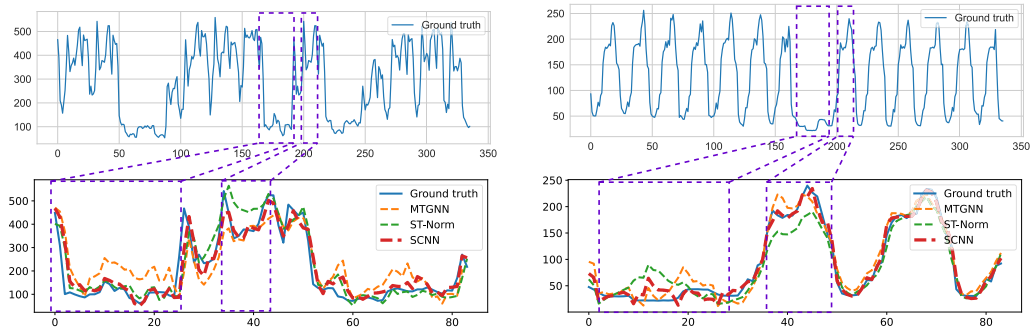


Figure. 5.10 Comparison of robustness.

demonstrates that shallow layers work on coarse-grained prediction, and deep layers perform fine-grained calibration by capturing the detailed changes presented in the MTS data. Fig. 5.9b shows that the prediction error of SCNN firstly decreases and then increases as the number of hidden channels increases. The number of input steps can affect the estimation of the long-term component and the seasonal component, thereby leading to differences in the accuracy of the forecast, as illustrated in Fig. 5.9c. It is appealing to find from Fig. 5.9d that SCNN behaves competitively with the kernel of size 1, which means that the correlations across the local observations vanish once conditioned on the set of structured components. Fig. 5.9e and Fig. 5.9f demonstrate the effectiveness of the setup of the other two hyper-parameters.

5.4.6 Robustness

To evaluate model robustness, we subject each model to two commonly encountered data corruptions: i.i.d. Gaussian noise and missing data. The less a model's performance degrades in the presence of these corruptions, the more robust it can be considered. In our comparison, we include SCNN, SCNN w/o aux, SCINET,



(a) (b)

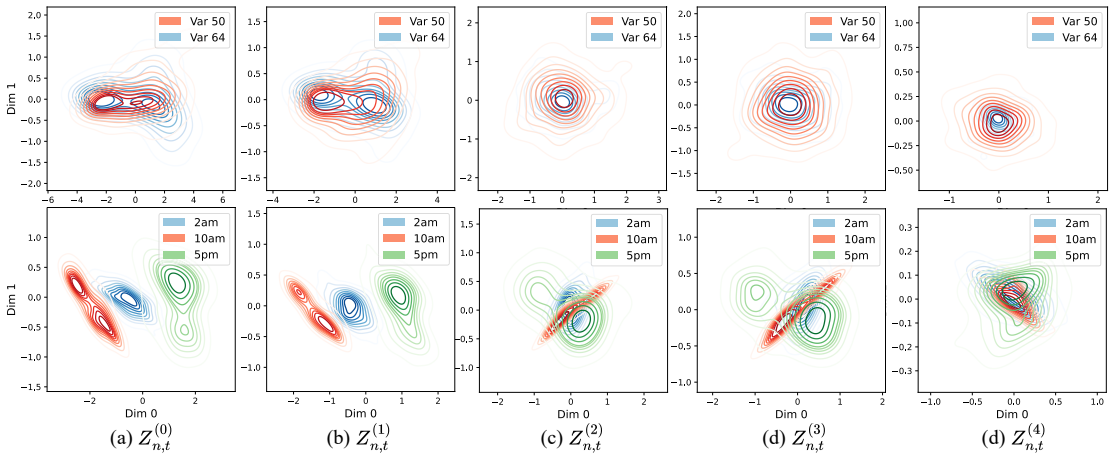
Figure. 5.11 Case Studies. The results demonstrate that the SCNN consistently achieves the lowest prediction error among the three models in diverse and challenging scenarios of distribution shifts and anomalies.

GW, MTGNN, and AGCRN, with 'SCNN w/o aux' denoting the SCNN model without the structural regularization module enabled.

As demonstrated in Fig. 5.10, SCNN consistently exhibits the smallest performance degradation among all models under each type of corruption. This is true even when compared to SCNN w/o aux, which underlines the important role of the structural regularization module in enhancing SCNN's robustness. These results underscore SCNN's superior robustness relative to the other models examined, highlighting its resilience in the face of data corruption.

5.4.7 Case Studies

We provide evidence through two case studies that the SCNN consistently outperforms two competitive baselines, MTGNN and ST-Norm, particularly when dealing with anomalous patterns. This is illustrated in Fig. 5.11. The left figure represents an episode of a time series demonstrating irregular behavior, while the right figure



(a) $Z_{n,t}^{(0)}$ (b) $Z_{n,t}^{(1)}$ (c) $Z_{n,t}^{(2)}$ (d) $Z_{n,t}^{(3)}$ (d) $Z_{n,t}^{(4)}$

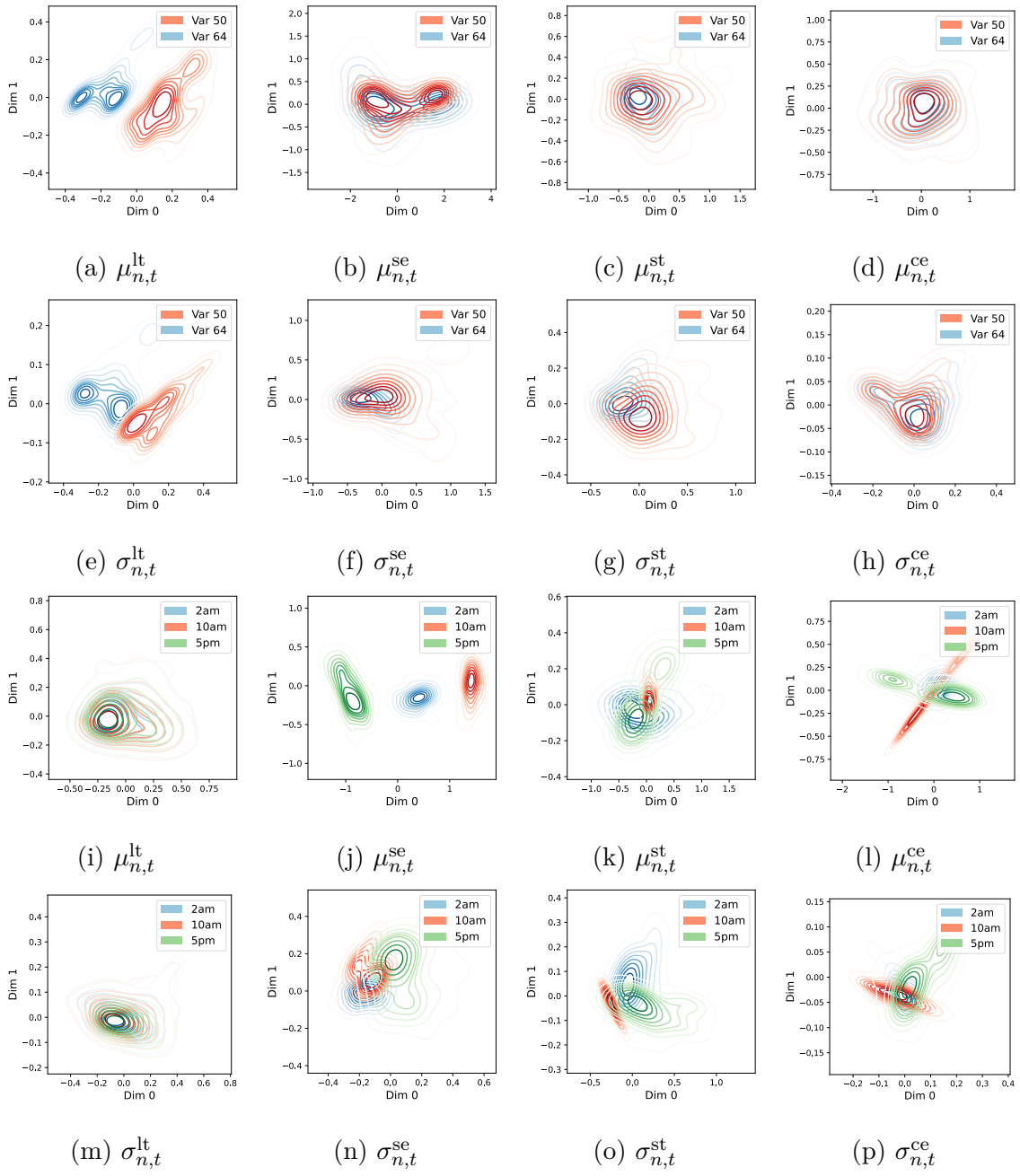
Figure. 5.12 Visualization of residual representations.

exhibits another episode characterized by a distinct and primarily regular daily cycle.

In examining both regular and irregular episodes, we focus on two specific periods and plot the rolling predictions—predictions made on a rolling basis using a sliding window of data—for the initial forecast horizon as generated by the three models during these periods. The results demonstrate that the SCNN consistently achieves the lowest prediction error among the three models in all four scenarios. This indicates the efficacy of our design in enabling the SCNN to effectively handle anomalies or distribution shifts in a variety of contexts. These results underscore the potential of SCNN to deliver reliable and robust forecasting in diverse and challenging scenarios.

5.4.8 Qualitative Study

We conduct a qualitative study to cast light on how the structure of representation space is progressively reshaped by iteratively disentangling the structured components. The structured components are visualized in Fig. 5.13. For the



(a) $\mu^{\text{lt}}_{n,t}$ (b) $\mu^{\text{se}}_{n,t}$ (c) $\mu^{\text{st}}_{n,t}$ (d) $\mu^{\text{ce}}_{n,t}$

(e) $\sigma^{\text{lt}}_{n,t}$ (f) $\sigma^{\text{se}}_{n,t}$ (g) $\sigma^{\text{st}}_{n,t}$ (h) $\sigma^{\text{ce}}_{n,t}$

(i) $\mu^{\text{lt}}_{n,t}$ (j) $\mu^{\text{se}}_{n,t}$ (k) $\mu^{\text{st}}_{n,t}$ (l) $\mu^{\text{ce}}_{n,t}$

(m) $\sigma^{\text{lt}}_{n,t}$ (n) $\sigma^{\text{se}}_{n,t}$ (o) $\sigma^{\text{st}}_{n,t}$ (p) $\sigma^{\text{ce}}_{n,t}$

Figure. 5.13 Visualization of structured components.

sake of visualization, we apply principal component analysis (PCA) to obtain the two-dimensional embeddings of the residual representations. Then, to convey the characteristics of the structure for any component, we perform two coloring schemes,

where the first scheme, as shown in the first row of Fig. 5.12, separates the data points according to their spatial identities, and the second one, displayed in the second row of Fig. 5.12, respects their temporal identities. For clarity, we plot the kernel density estimate (KDE) for each group of points. It is conspicuous that by progressively removing the structured components from $Z_{n,t}^{(0)}$, the residual representations with different spatial and temporal identities gradually align together, suggesting that the distinct structural information has been held by the structured components.

Chapter 6

Conclusion and Future Work

In the intricate domain of time series forecasting, the challenges and nuances are manifold. This thesis has delved deep into these complexities, presenting a series of comprehensive studies that collectively offer transformative solutions for multivariate time series forecasting. Each chapter has contributed distinct insights, methodologies, and frameworks, culminating in a holistic understanding of the field.

Chapter 3 embarked on the pivotal task of forecasting urban travel demand, emphasizing the importance of considering multiple transportation modes. Unlike traditional studies that focus on singular modes, our approach embraced the co-evolving dynamics inherent in various transportation channels. The chapter's hallmark was the introduction of a co-evolving pattern learning framework, a pioneering model tailored to decode the intricate space-time-varying inter-dependencies. This model's ability to discern dynamic correlations across transportation modes, while capturing the nuanced interplay of demand influenced by both time and location, underscores its significance in urban mobility forecasting.

Chapter 4 shifted the lens to the dynamic patterns observed across spatial and temporal dimensions in time series data. By dissecting determinants like population density, time of day, and weather conditions, the chapter illuminated the

space-time-varying distribution of data. Our approach's novelty lay in its ability to extract determinants directly from time series data, integrating them into our model. The introduction of a specialized module to differentiate between low and high-frequency components further showcased our commitment to capturing both the broader and specific patterns of space-time-varying distribution.

Chapter 5 presented the Structured Component-based Neural Network (SCNN), a groundbreaking solution in the realm of MTS forecasting. This chapter's exploration unveiled the dynamic shifts in distribution and autocorrelation, emphasizing the need for models that can adapt to these shifts. The SCNN, with its capability to decouple structured components of MTS data, emerged as a beacon of innovation. Its adaptability, scalability, and interpretability set it apart, promising a new benchmark in time series analysis.

In summation, this thesis has journeyed through the multifaceted world of time series forecasting, presenting innovative solutions at each turn. From understanding urban mobility's co-evolving dynamics to delving into the space-time-varying distribution of data, and finally, introducing the SCNN, a model that promises adaptability, scalability, and interpretability, the contributions have been profound. As we conclude, it's evident that the future of multivariate time series forecasting is on the cusp of a transformative era, and this thesis stands as a testament to that impending revolution.

As we gaze into the horizon, three promising avenues beckon:

Automated Neural Architecture Identification: We aspire to automate the discernment of optimal neural architectures, utilizing the foundational modules and operations we've discussed as cornerstones. This endeavor could revolutionize the tedious manual process of pinpointing the best architecture for each unique

dataset. By doing so, we might unveil the intricate structures and meta-knowledge embedded within time-series data. This exploration could potentially offer profound insights into the very essence of time-series analysis.

Heterogeneous Time Series Relationships: Our goal is to dissect the multifaceted relationships within time series at the component level. Recognizing that time series data fragments into components with varied dynamics, we postulate that these components might exhibit distinct inter-dependencies. This insight paves the way for a deeper understanding of the intricate relationships between series pairs, urging us to develop methods that can adeptly decode these inter-dependencies.

Addressing Irregular Components in Forecasting: One of the paramount challenges in time series forecasting is the accurate emulation of erratic components. While normalization techniques aid in distinguishing these from their regular counterparts, replicating their dynamics remains a formidable challenge. Given the volatile nature of these components, there's a scarcity of consistent patterns in datasets. We believe that curriculum learning could be the linchpin solution, enabling us to pinpoint challenging instances and employ a progressive "hard to easy" training strategy.

References

[1] Alon, U. and Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*.

[2] BAI, L., Yao, L., Li, C., Wang, X., and Wang, C. (2020). Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems*, 33.

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

[4] Bandara, K., Bergmeir, C., and Hewamalage, H. (2021). Lstm-msnet: Leveraging forecasts on sets of related time series with multiple seasonal patterns. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4):1586–1599.

[5] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[6] Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, Y., Tong, J., et al. (2020). Spectral temporal graph neural network for multivariate time-series forecasting. *Proceedings of the NeurIPS 2020*.

[7] Chai, D., Wang, L., and Yang, Q. (2018). Bike flow prediction with multi-graph convolutional networks. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 397–400.

[8] Challu, C., Olivares, K. G., Oreshkin, B. N., Ramirez, F. G., Canseco, M. M., and Dubrawski, A. (2023). Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6989–6997.

[9] Chen, C., Li, K., Teo, S. G., Zou, X., Li, K., and Zeng, Z. (2020). Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(4):1–23.

[10] Chen, C., Li, K., Teo, S. G., Zou, X., Wang, K., Wang, J., and Zeng, Z. (2019). Gated residual recurrent graph neural networks for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 485–492.

[11] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

[12] Choi, J., Choi, H., Hwang, J., and Park, N. (2022). Graph neural controlled differential equations for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[13] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[14] Cirstea, R.-G., Yang, B., Guo, C., Kieu, T., and Pan, S. (2022). Towards spatio-temporal aware traffic time series forecasting. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2900–2913.

[15] Cook, R. D. and Weisberg, S. (1983). Diagnostics for heteroscedasticity in regression. *Biometrika*, 70(1):1–10.

[16] Das, A., Kong, W., Leach, A., Sen, R., and Yu, R. (2023). Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*.

[17] Deng, J., Chen, X., Fan, Z., Jiang, R., Song, X., and Tsang, I. W. (2021a). The pulse of urban transport: exploring the co-evolving pattern for spatio-temporal forecasting. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(6):1–25.

[18] Deng, J., Chen, X., Jiang, R., Song, X., and Tsang, I. W. (2021b). St-norm: Spatial and temporal normalization for multi-variate time series forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 269–278.

[19] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[20] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[21] Ding, D., Zhang, M., Pan, X., Yang, M., and He, X. (2019). Modeling extreme events in time series prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1114–1122.

[22] Engle, R. (2002). Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics*, 20(3):339–350.

[23] Fan, W., Wang, P., Wang, D., Wang, D., Zhou, Y., and Fu, Y. (2023). Dish-ts: A general paradigm for alleviating distribution shift in time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7522–7529.

[24] Fan, W., Zheng, S., Yi, X., Cao, W., Fu, Y., Bian, J., and Liu, T.-Y. (2022). DEPTS: Deep expansion learning for periodic time series forecasting. In *International Conference on Learning Representations*.

[25] Fang, S., Zhang, Q., Meng, G., Xiang, S., and Pan, C. (2019). Gstnet: Global spatial-temporal network for traffic flow prediction. In *Proceedings of the Twenty-*

Eighth International Joint Conference on Artificial Intelligence, IJCAI, pages 10–16.

[26] Feng, J., Li, Y., Lin, Z., Rong, C., Sun, F., Guo, D., and Jin, D. (2021). Context-aware spatial-temporal neural network for citywide crowd flow prediction via modeling long-range spatial dependency. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(3):1–21.

[27] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37.

[28] 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 *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3656–3663.

[29] Gong, Y., Li, Z., Zhang, J., Liu, W., Zheng, Y., and Kirsch, C. (2018). Network-wide crowd flow prediction of sydney trains via customized online non-negative matrix factorization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1243–1252.

[30] Guo, K., Hu, Y., Sun, Y., Qian, S., Gao, J., and Yin, B. (2021a). Hierarchical graph convolution networks for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 151–159.

[31] Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. (2019a). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 922–929.

[32] Guo, S., Lin, Y., Li, S., Chen, Z., and Wan, H. (2019b). Deep spatial–temporal 3d convolutional neural networks for traffic data forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3913–3926.

[33] Guo, S., Lin, Y., Li, S., Chen, Z., and Wan, H. (2019c). Deep spatial–temporal 3d convolutional neural networks for traffic data forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3913–3926.

[34] Guo, S., Lin, Y., Wan, H., Li, X., and Cong, G. (2021b). Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5415–5428.

[35] Han, L., Ye, H.-J., and Zhan, D.-C. (2023). The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting.

[36] Harvey, A. C. (1976). Estimating regression models with multiplicative heteroscedasticity. *Econometrica: Journal of the Econometric Society*, pages 461–465.

[37] He, S. and Shin, K. G. (2020). Towards fine-grained flow forecasting: A graph attention approach for bike sharing systems. In *Proceedings of The Web Conference 2020*, pages 88–98.

[38] Hillmer, S. C. and Tiao, G. C. (1982). An arima-model-based approach to seasonal adjustment. *Journal of the American Statistical Association*, 77(377):63–70.

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

[40] Holtz-Eakin, D., Newey, W., and Rosen, H. S. (1988). Estimating vector autoregressions with panel data. *Econometrica: Journal of the econometric society*, pages 1371–1395.

[41] Huang, C., Zhang, C., Zhao, J., Wu, X., Yin, D., and Chawla, N. (2019). Mist: A multiview and multimodal spatial-temporal learning framework for citywide abnormal event forecasting. In *The World Wide Web Conference*, WWW '19, page 717–728, New York, NY, USA. Association for Computing Machinery.

[42] Jiang, R., Song, X., Huang, D., Song, X., Xia, T., Cai, Z., Wang, Z., Kim, K.-S., and Shibasaki, R. (2019). Deepurbanevent: A system for predicting citywide crowd dynamics at big events. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2114–2122. ACM.

[43] Jiang, R., Wang, Z., Yong, J., Jeph, P., Chen, Q., Kobayashi, Y., Song, X., Fukushima, S., and Suzumura, T. (2023). Spatio-temporal meta-graph learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8078–8086.

[44] Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. (2021). Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

[45] Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. (2022). Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

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

[47] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

[48] Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., et al. (2021). Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR.

[49] Lai, G., Chang, W.-C., Yang, Y., and Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104.

[50] Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., et al. (2022). Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*.

[51] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[52] Li, L., Pagnucco, M., and Song, Y. (2022). Graph-based spatial transformer with memory replay for multi-future pedestrian trajectory prediction. In *Proceed-*

ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2231–2241.

[53] Li, M., Qin, Z., Jiao, Y., Yang, Y., Wang, J., Wang, C., Wu, G., and Ye, J. (2019a). Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference*, pages 983–994.

[54] Li, Q., Han, Z., and Wu, X.-M. (2018a). Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[55] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019b). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, pages 5243–5253.

[56] Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018b). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*.

[57] Li, Y., Zhu, Z., Kong, D., Xu, M., and Zhao, Y. (2019c). Learning heterogeneous spatial-temporal representation for bike-sharing demand prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1004–1011.

[58] Liang, Y., Ke, S., Zhang, J., Yi, X., and Zheng, Y. (2018). Geoman: Multi-level attention networks for geo-sensory time series prediction. In *IJCAI*, pages 3428–3434.

[59] Lin, S., Lin, W., Wu, W., Wang, S., and Wang, Y. (2023). Petformer: Long-term time series forecasting via placeholder-enhanced transformer.

[60] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1020–1027.

[61] Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., and Xu, Q. (2022a). Scinet: Time series modeling and forecasting with sample convolution and interaction. *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS), 2022*.

[62] Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. (2022b). Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*.

[63] Liu, Y., Wu, H., Wang, J., and Long, M. (2022c). Non-stationary transformers: Rethinking the stationarity in time series forecasting. *arXiv preprint arXiv:2205.14415*.

[64] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., and Zhang, G. (2018). Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363.

[65] Ma, X., Zhou, C., Kong, X., He, J., Gui, L., Neubig, G., May, J., and Zettlemoyer, L. (2023). Mega: Moving average equipped gated attention. In *The Eleventh International Conference on Learning Representations*.

[66] 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*, pages 3111–3119.

[67] Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2023). A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*.

[68] Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2020). N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*.

[69] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

[70] Pan, B., Demiryurek, U., and Shahabi, C. (2012). Utilizing real-world transportation data for accurate traffic prediction. In *2012 IEEE 12th International Conference on Data Mining*, pages 595–604. IEEE.

[71] 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 *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1720–1730.

[72] Pan, Z., Zhang, W., Liang, Y., Zhang, W., Yu, Y., Zhang, J., and Zheng, Y. (2020). Spatio-temporal meta learning for urban traffic prediction. *IEEE Transactions on Knowledge and Data Engineering*.

[73] Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2020). Deep adaptive input normalization for time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3760–3765.

[74] Patton, A. J. (2012). A review of copula models for economic time series. *Journal of Multivariate Analysis*, 110:4–18.

[75] Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in neural information processing systems*, pages 7785–7794.

[76] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1985). Learning internal representations by error propagation.

[77] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2019). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*.

[78] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191.

[79] Shabani, M. A., Abdi, A. H., Meng, L., and Sylvain, T. (2023). Scaleformer: Iterative multi-scale refining transformers for time series forecasting. In *The Eleventh International Conference on Learning Representations*.

[80] Shang, C., Chen, J., and Bi, J. (2021). Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*.

[81] Shao, Z., Zhang, Z., Wang, F., Wei, W., and Xu, Y. (2022). Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4454–4458.

[82] Shi, X., 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. *Advances in neural information processing systems*, 28.

[83] Shumway, R. H., Stoffer, D. S., and Stoffer, D. S. (2000). *Time series analysis and its applications*, volume 3. Springer.

[84] Tobler, W. R. (1970). A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240.

[85] Tong, Y., Chen, Y., Zhou, Z., Chen, L., Wang, J., Yang, Q., Ye, J., and Lv, W. (2017). The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1653–1662.

[86] Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. (2016). Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022.

[van den Oord et al.] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125.

[88] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[89] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. *International Conference on Learning Representations*. accepted as poster.

[90] Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., and Xiao, Y. (2023a). MICN: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*.

[91] Wang, Y., Gao, Z., Long, M., Wang, J., and Philip, S. Y. (2018). Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *International Conference on Machine Learning*, pages 5123–5132. PMLR.

[92] Wang, Y., Long, M., Wang, J., Gao, Z., and Yu, P. S. (2017). Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. *Advances in neural information processing systems*, 30.

[93] Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., and Januschowski, T. (2019a). Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617. PMLR.

[94] Wang, Y., Wu, H., Zhang, J., Gao, Z., Wang, J., Philip, S. Y., and Long, M. (2022a). Predrnn: A recurrent neural network for spatiotemporal predictive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2208–2225.

[95] Wang, Y., Yin, H., Chen, H., Wo, T., Xu, J., and Zheng, K. (2019b). Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1227–1235.

[96] Wang, Z., Nie, Y., Sun, P., Nguyen, N. H., Mulvey, J., and Poor, H. V. (2023b). St-mlp: A cascaded spatio-temporal linear framework with channel-independence strategy for traffic forecasting. *arXiv preprint arXiv:2308.07496*.

[97] Wang, Z., Xu, X., Trajcevski, G., Zhang, W., Zhong, T., and Zhou, F. (2022b). Learning latent seasonal-trend representations for time series forecasting. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[98] Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. (2021). Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International Conference on Learning Representations*.

[99] Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. (2022a). Deeptime: Deep time-index meta-learning for non-stationary time-series forecasting. *arXiv preprint arXiv:2207.06046*.

[100] Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. (2022b). Etsformer: Exponential smoothing transformers for time-series forecasting.

[101] Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. (2023). Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*.

[102] Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

[103] Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., and Zhang, C. (2020). Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 753–763.

[104] Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. (2019). Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 1907–1913. AAAI Press.

[105] Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G.-J., and Xiong, H. (2020). Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*.

[106] Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., Liu, C., Bian, W., and Ye, J. (2018). Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 905–913.

[107] Xue, W., Zhou, T., Wen, Q., Gao, J., Ding, B., and Jin, R. (2023). Make transformer great again for time series forecasting: Channel aligned robust dual transformer.

[108] Yang, S., Liu, J., and Zhao, K. (2021). Space meets time: Local spacetime neural network for traffic flow forecasting. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 817–826. IEEE.

[109] Yao, H., Liu, Y., Wei, Y., Tang, X., and Li, Z. (2019a). Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*, pages 2181–2191.

[110] Yao, H., Tang, X., Wei, H., Zheng, G., and Li, Z. (2019b). Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5668–5675.

[111] 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 *Thirty-Second AAAI Conference on Artificial Intelligence*.

[112] Ye, J., Sun, L., Du, B., Fu, Y., Tong, X., and Xiong, H. (2019). Co-prediction of multiple transportation demands based on deep spatio-temporal neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 305–313. ACM.

[113] Yu, B., Yin, H., and Zhu, Z. (2018). Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640.

[114] Yu, C., Wang, F., Shao, Z., Sun, T., Wu, L., and Xu, Y. (2023). Dsformer: A double sampling transformer for multivariate time series long-term prediction.

[115] Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2022). Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*.

[116] Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2023). Are transformers effective for time series forecasting?

[117] Zhang, J., Zheng, Y., and Qi, D. (2017). Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[118] Zhang, J., Zheng, Y., Qi, D., Li, R., and Yi, X. (2016). Dnn-based prediction model for spatio-temporal data. In *Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 1–4.

[119] Zhang, J., Zheng, Y., Sun, J., and Qi, D. (2019). Flow prediction in spatio-temporal networks based on multitask deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):468–478.

[120] Zhang, W., Liu, H., Liu, Y., Zhou, J., Xu, T., and Xiong, H. (2020). Semi-supervised city-wide parking availability prediction via hierarchical recurrent graph neural network. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3984–3996.

[121] Zhang, X., Huang, C., Xu, Y., Xia, L., Dai, P., Bo, L., Zhang, J., and Zheng, Y. (2021). Traffic flow forecasting with spatial-temporal graph diffusion network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15008–15015.

[122] Zhang, X., Jin, X., Gopalswamy, K., Gupta, G., Park, Y., Shi, X., Wang, H., Maddix, D. C., and Wang, B. (2022). First de-trend then attend: Rethinking attention for time-series forecasting. In *NeurIPS'22 Workshop on All Things Attention: Bridging Different Perspectives on Attention*.

[123] Zhang, Y. and Yan, J. (2023). Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*.

[124] Zhang, Z., Wang, X., and Gu, Y. (2023). Sageformer: Series-aware graph-enhanced transformers for multivariate time series forecasting.

[125] Zhao, J., Huang, F., Lv, J., Duan, Y., Qin, Z., Li, G., and Tian, G. (2020). Do rnn and lstm have long memory? In *International Conference on Machine Learning*, pages 11365–11375. PMLR.

[126] Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., and Li, H. (2019). T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858.

[127] Zheng, C., Fan, X., Wang, C., and Qi, J. (2020a). Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1234–1241.

[128] Zheng, H., Lin, F., Feng, X., and Chen, Y. (2020b). A hybrid deep learning model with attention-based conv-lstm networks for short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 22(11):6910–6920.

[129] 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 AAAI*.

[130] Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. (2022). Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pages 27268–27286. PMLR.

[131] Zhou, X., Shen, Y., Zhu, Y., and Huang, L. (2018). Predicting multi-step citywide passenger demands using attention-based neural networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 736–744.

[132] Zhou, Z.-H. (2022). Open-environment machine learning. *National Science Review*, 9(8):nwac123.

[133] Žliobaitė, I., Pechenizkiy, M., and Gama, J. (2016). An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, pages 91–114.

[134] Zonoozi, A., Kim, J.-j., Li, X.-L., and Cong, G. (2018). Periodic-crn: A convolutional recurrent model for crowd density prediction with recurring periodic patterns. In *IJCAI*, pages 3732–3738.