UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Spatial-Temporal Representation Learning for Traffic Forecasting

by Changlu Chen

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Ling Chen, Yanbin Liu and Chengqi Zhang

University of Technology Sydney
Faculty of Engineering and Information Technology

March 2024

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Changlu Chen, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering Information and 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.

Production Note:
Signature removed prior to publication.

01/03/2024

ABSTRACT

Spatial-Temporal Representation Learning

for Traffic Forecasting

by

Changlu Chen

Traffic forecasting is the cornerstone of the Intelligent Transportation System (ITS). Accurate traffic prediction can effectively promote traffic management and urban planning. This task is challenging due to the dynamic and complex spatial-temporal correlations in the time-evolving traffic network. Spatially, the traffic conditions of nearby locations have a dynamic influence on each other. Temporally, the traffic conditions exhibit elusive patterns due to various external factors such as weather, rush hour, weekends, holidays, etc. With the development of deep learning, spatial-temporal representation learning has become the mainstream approach to traffic forecasting tasks. This thesis investigates techniques for learning effective spatial-temporal representation for traffic forecasting systems. The exploration is motivated by the following challenges that could hinder the further development of spatial-temporal representation learning for enhancing the final forecasting performance.

1. The diverse complexity of spatial-temporal representation learning. The complexity of diverse forecasting tasks is non-uniformly distributed across various spatial locations (e.g. suburb vs. downtown) and temporal steps (e.g. rush hour vs. off-peak).

2. The data scarcity and imbalance in traffic forecasting datasets. For example, the temporal observations of traffic accidents exhibit ultra-rareness due to the inherent properties of accident occurrences. This leads to the severe scarcity of risk samples for comprehensive representation learning.

3. The under-exploration of frequency domain spatial-temporal representation learning. The spectral features could provide potential compensation for the time domain representation learning.

4. Distribution shift in the non-stationary traffic data. Despite the recent success of deep neural networks in spatial-temporal forecasting, existing methods suffer from distribution shifts between the training and test data, failing to address the non-stationary and abrupt changes at test time.

To solve the above challenges, this thesis proposes four models for various traffic forecasting applications, ranging from traffic flow, traffic accident, and traffic demand forecasting. All of these methods aim to explore challenging spatial-temporal representation learning for enhancing traffic forecasting performance. Specifically, we first propose a Bidirectional Spatial-Temporal Adaptive Transformer (Bi-STAT) for accurate traffic flow forecasting, which devises the recurrent mechanism with a novel Dynamic Halting Module (DHM) to dynamically learn the spatial-temporal representation in the traffic streams according to their complexities. Secondly, we propose a contrastive learning approach with the multi-kernel networks, to learn the traffic accident representation under temporal scarcity and imbalanced spatial distribution. Then, we design a novel embedded 2D spectral learning framework to explore the traffic features in the frequency domain. Lastly, we propose a novel test-time training framework for spatial-temporal representation learning to alleviate the distribution shift in traffic data.

We conduct comprehensive experiments with real-world traffic forecasting datasets to corroborate the superiority of all the models over the state-of-the-art baselines, and also with extensive ablation studies to demonstrate the effectiveness of the different modules.

Dissertation directed by Professor Ling Chen

AAII - Australian Artificial Intelligence Institute, UTS

Dedication

To my parents Xiaolian Chang and Weili Chen, my sister, Changtong Chen.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Ling Chen for the continuous support of my whole Ph.D. study. She is such a nice person with a good heart. Her sincere and plain attitude towards scientific research and life has been a tremendous source of motivation for me. The simplicity and purity of her character have profoundly impacted me, serving as a perpetual reminder to prioritize what truly matters in every endeavor and circumstance.

I would also like to thank my master's supervisor, Prof. Kun Zhan, for guiding me on the academic journey and presenting me with this invaluable opportunity. Without his support, I might have worked in a state-owned company with limited financial prospects. I appreciate him for opening the door to a future world filled with challenges and, equally, opportunities for me.

Then, I would like to show my serious respect and appreciation to my co-supervisor, Dr. Yanbin Liu, who is the one who gives me enormous help in all of my research projects. Without his help, I would not have been able to get a chance to complete my Ph.D. degree and sit here to write this acknowledgment. I am so grateful to his weekly guidance, always being patient with my problems, and tolerant of my slow pace and naive mistakes. I have learned a lot of techniques from him, which would benefit me throughout my lifetime, regardless of the path I choose.

Finally, I would like to thank my family – my mother and father, who actually know nothing about my research, but are still proud of me even though I am not so good at what I am doing right now. I also want to express my gratitude to my partner Chaoxi Niu, who always provides me with selfless love and continuous support in my life and research. Their enduring companionship has always been a constant source of strength in my life and career.

Changlu Chen

Sydney, Australia, 06/02/2024.

List of Publications

1. **Chen, C.**, Liu, Y., Chen, L., Zhang, C. (2022). Bidirectional spatial-temporal adaptive transformer for Urban traffic flow forecasting. IEEE Transactions on Neural Networks and Learning Systems. (Core A*)

2. **Chen, C.**, Liu, Y., Chen, L., Zhang, C. (2023). RiskContra: A Contrastive Learning Approach to Traffic Accident Forecasting with Multi-Kernel Networks. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD). (Core A)

3. **Chen, C.**, Liu, Y., Chen, L., Zhang, C. (2024). Test-Time Training for Spatial-Temporal Forecasting. In Proceedings of the 2023 SIAM International Conference on Data Mining (SDM). (Core A)

4. **Chen, C.**, Liu, Y., Chen, L., Zhang, C. (2024). Multivariate Traffic Demand Prediction via 2D Spectral Learning and Global Spatial Optimization. ECML-PKDD. (Under Review) (Core A)

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

1.1.1 Urban Traffic Forecasting

The rapid growth of urbanization and modernization has propelled us into the era of smart cities, wherein Intelligent Transportation System (ITS) plays an indispensable role. The advancement of mobile internet and position technology has significantly increased the accessibility of extensive traffic data, making traffic forecasting an essential component of ITS [116].

Accurate traffic forecasting provides practical insights that are crucial to intelligent urban planning and effective traffic management, with applications spanning a diverse range of scenarios. For instance, foreseeing traffic flow in advance facilitates seamless route planning, helping avoid heavy congestion during commuting. Accurate traffic demand forecasting is paramount for car-sharing companies, enabling them to allocate resources efficiently to high-demand areas and curbing potential resource waste. Additionally, traffic accident forecasting plays a crucial role in preventing possible disasters, safeguarding lives, and protecting properties. The exploration and anticipation of future traffic conditions is of great significance not only for public safety stakeholders but also for transportation administrators, individual drivers, and travelers alike.

Traffic forecasting aims to predict traffic conditions (e.g. traffic flow, traffic demand, traffic accident, etc) at the target spatial locations and future temporal steps based on historical observations, which are recorded by both physical and

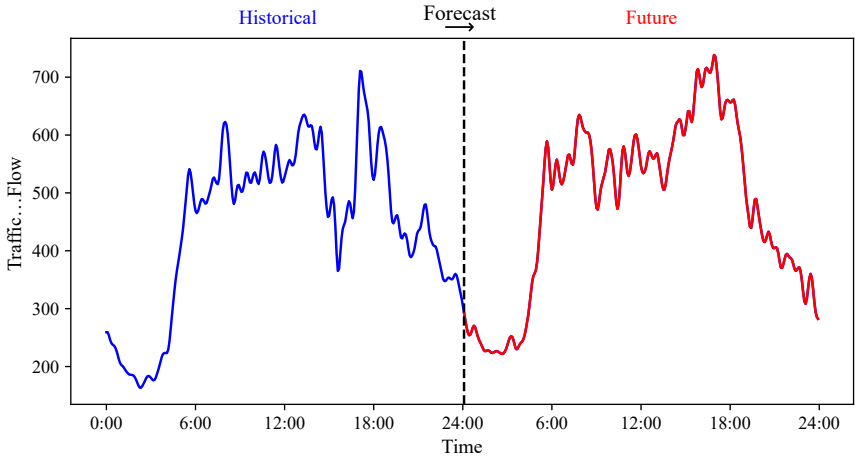


Figure 1.1 : Traffic forecasting aims to predict future traffic conditions from historical observations.

virtual sensors spreading across the target areas. It is a very challenging task because traffic data is spatial-temporal data consisting of spatial correlations from various locations and temporal correlations from different time steps. The space and time dimensions are intertwined with each other, leading to the complex and dynamic spatial-temporal correlations, as shown in Figure 1.2.

Spatially, different locations have various dependencies with each other, where the nearby regions might tend to have close relationships, while the geographically distant regions might also have high correlations due to similar semantic functionality. For the same location pairs, the spatial correlations among them dynamically change with time evolving. For example, the correlations between the working place and the apartment are different in the morning and evening.

Temporally, traffic data have inherent periodicity and show diverse patterns according to the peak or off-peak hours, holidays, etc. The short-term temporal correlations from chronologically close time steps can have a strong influence on the target future time step. Meanwhile, the long-term temporal correlations can also be critical for future forecasting because of the time delay in some traffic states. For

example, the impact of traffic accidents or congestion can last over a long range of future time steps, which is an important factor for temporal correlation modeling. Besides, the traffic conditions exhibit elusive patterns due to various external factors such as weather information, Point of Interests, road construction, etc. The

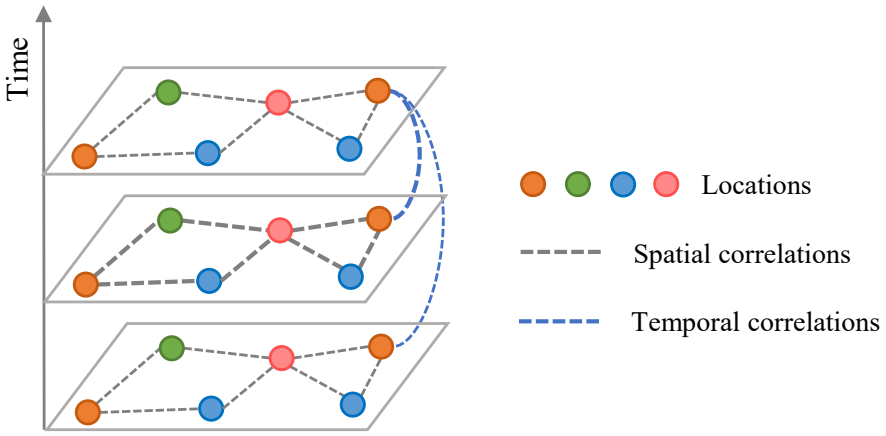


Figure 1.2 : The correlations between different locations and different time steps are dynamic.

core assumption behind spatial-temporal traffic forecasting is that a location's future conditions are influenced by its historical conditions as well as its neighbors' historical conditions. Thus, spatial-temporal representation learning is an essential component in traffic forecasting.

In early studies, statistical models such as Historical Average (HA) [101], autoregressive integrated moving average (ARIMA) [102] and VAR [133] and their variants [59, 69, 83] are proposed to perform the traffic forecasting task. However, these traditional methods can not achieve competitive forecasting performance due to the following factors:

- They make stationary and independent assumptions on traffic data, thus failing to capture the dynamic and inter-correlated relations in real-world traffic data.

- They exploit limited features of accident data, ignoring various contextual factors that contribute to the diverse traffic patterns.
- They mainly concentrate on univariate data processing, while in real-world tasks, there are more spatial-temporal data with multiple input variables.
- They focus on short-term forecasting problems with most of them only dealing with one-step forecasts, which can not perform the longer-term forecasting with multi-horizon ahead of the present time step.

1.1.2 Spatial-Temporal Representation Learning

Spatial-temporal representation learning is of critical importance to traffic forecasting tasks. The emerging deep learning techniques provide a promising alternative to urban traffic forecasting, due to its capability to extract discriminative features and capture complicated dynamic correlations. Therefore, an assortment of deep network architectures have been applied to learn the complex spatial-temporal correlations in traffic sequences. Specifically, Convolutional Neural Networks (CNNs) [53] can capture the detailed and local spatial-temporal features by processing and aggregating the context information from a small neighborhood. However, the limited receptive field restrains them from capturing long-range spatial-temporal dependencies. Graph Convolutional Networks (GCNs) [51] have been pervasively applied to capturing the complex dependencies in multivariate spatial-temporal data. However, they suffer from the inherently static graph architecture, which prevents them from developing time-evolving dependencies in traffic networks. Recurrent Neural Networks (RNNs) [20, 88] are naturally powerful for learning the sequential features in traffic data, but they suffer from error propagation issues due to the reliance on the last hidden state. Furthermore, Transformers [91] can effectively capture the long-range dependencies benefiting from the self-attention mechanism, which dynamically attends to each position in the

sequence and aggregates the most contributing features to generate the final representation. Nonetheless, the blind similarity comparison has caused an unnecessary computation burden, which is prohibitively expensive for long sequences.

1.2 Research Objectives

This thesis aims to promote spatial-temporal representation learning to enhance the performance of urban traffic forecasting. The quality of spatial-temporal representation plays a pivotal role in the performance of traffic forecasting. Specifically, the spatial correlations among different locations in the traffic network, the temporal correlations among multiple time steps of different ranges, as well as the intertwined spatial-temporal dependencies that can be captured during the spatial-temporal representation learning are the most contributing factors for future forecasting. This thesis is motivated by the following challenges in spatial-temporal representation learning, which hinder existing methods from becoming more efficient and accurate traffic condition predictors:

- Firstly, the diverse complexity in spatial-temporal representation learning is non-uniformly distributed across various spatial locations (e.g. suburb vs. downtown) and temporal steps (e.g. rush hour vs. off-peak). For example, spatially, traffic flow in urban areas witnesses higher values and more complex patterns compared to that of rural areas, which shows simpler and smoother patterns. Temporally, the rush hour flow, oscillating frequently, is harder to predict than the off-peak traffic patterns with a stable and simple pattern. If the tasks with diverse complexities are dealt with the same learning strategy, the detailed and fluctuated patterns will not be fully captured, while the simple ones might face the overfitting problem and excess computing resources, which can hinder the further development of spatial-temporal forecasting learning.

To address this challenge, we propose a Bidirectional Spatial-Temporal Adaptive Transformer (Bi-STAT) for accurate traffic forecasting. Bi-STAT adopts an encoder-decoder architecture, where both the encoder and the decoder maintain a spatial-adaptive Transformer and a temporal-adaptive Transformer structure. A recurrent mechanism with a novel Dynamic Halting Module (DHM) is proposed to incorporate into each Transformer to dynamically process the traffic streams according to their task complexities. In this way, this model can effectively enhance spatial-temporal representation learning by assigning more computation resources to more complex patterns and less to simpler ones, which helps achieve a balance between performance and efficiency in traffic forecasting tasks.

- Secondly, the data scarcity due to the infrequent occurrences of specific traffic states such as traffic accidents brings a great challenge to spatial-temporal representation learning. For example, the dominant number of zeros in traffic accident data not only makes it difficult for the deep networks to be sufficiently trained, but it can also easily lead the representation learning biased to the non-accident forecasting result. Although some works have focused on dealing with this challenge, most of them use prior knowledge to design the strategy without incorporating it into spatial-temporal representation learning.

  To solve this challenge, this thesis designs a novel contrastive learning approach, which leverages the periodic patterns to derive a tailored mixup strategy for accident sample augmentation. This way, the contrastively learned features can better represent the accident samples, thus capturing higher-quality spatial-temporal representations for forecasting.

- Thirdly, although spatial-temporal representation learning has been extensively investigated in the time domain, frequency domain representation learn-

ing, which aims to learn the spectral features in the Fourier Transform data, is far less explored in the traffic forecasting domain. This gap in research may lead to an information loss, given that spatial-temporal data inherently comprises a time sequence with multiple frequency components. For example, the high-frequency component can represent the transient fluctuations while low frequencies are the smooth trend of traffic patterns. Besides, the spectral bias [7, 78] in the deep neural networks makes the high-frequency components of input variations underexplored.

To solve this challenge, we aim to design a model to explore the spatial-temporal representation learning in both the time and frequency domain, to make the spectral features complementary to the temporal features for better exploration of traffic data. Specifically, we design an embedded 2D spectral learning framework. Firstly, a well-devised spectral embedding function is employed to encapsulate both the low-frequency and high-frequency signals of the multivariate input. Secondly, we model the temporal variations and multivariate feature interactions as two effective dimensions in the frequency domain. The 2D Fourier transform is directly applied along both dimensions followed by Fourier domain representation learning to extract more intrinsic patterns for traffic forecasting.

- The final challenge is that, although extensive efforts are devoted to representation learning during the training stage, existing methods suffer from distribution shifts between the training and test data, failing to address the non-stationary and abrupt changes at test time. The influence of the distribution shifts in spatial-temporal representation learning is more severe due to the natural gap between the training and testing data caused by the temporal evolution.

To solve this challenge, we propose a novel test-time training framework for spatial-temporal forecasting. Instead of employing a fixed trained model, we adapt the trained model by learning directly from test data with potential shifts, which could mitigate the gap between the spatial-temporal representation in training and testing data to promote the forecasting results. To implement test-time training on spatial-temporal data, we devise a bidirectional cycle-consistent architecture consisting of a forward and a backward cyclic network. Each network has a shared encoder and two direction-aware decoders. At the test time, two self-supervised auxiliary tasks (forward→backward and backward→forward reconstruction) are proposed to adapt the trained model without accessing the target labels. We only access the first sample in the test data to avoid the information leak.

1.3 Contribution

The contributions of this thesis are summarized as follows:

- This is the first work to dynamically tackle the traffic forecasting problem according to the unique spatial-temporal complexity, which is realized by the proposed Bi-STAT method. We design a novel Dynamic Halting Module to parsimoniously assign the essential computation load for each task. It dynamically terminates the iterative running of the recurrent Transformer, thus being flexible and parameter-efficient. In contrast to existing methods, we simultaneously recollect past traffic conditions and predict future traffic conditions. This design is coherent with the brain mechanism and further improves the traffic forecasting performance.

- We propose the first attempt to apply contrastive learning to the traffic accident forecasting problem. By utilizing intrinsic periodic patterns of accident data, we integrate the mixup algorithm into the contrastive framework

seamlessly to address the temporal rareness challenge. We devise a delicate Multi-kernel CNN structure to explicitly capture the multi-granularity spatial correlations, thus addressing the spatial imbalance challenge.

- This is the first work to develop a 2D spectral learning model for multivariate spatial-temporal prediction, which can leverage spectral information from not only the temporal variations but also the multivariate feature interactions. We devise a novel spectral embedding function to capture diverse frequency components of the multivariate input data and thus alleviate the spectral bias of deep networks. We formulate spatial correlation learning as an optimization problem to exploit the global spatial correlations of the citywide traffic network.

- To the best of our knowledge, this is the first test-time training framework for spatial-temporal representation learning, which can adapt the trained model to unseen test data with unexpected distribution shifts. We propose a novel bidirectional cycle-consistent structure to perform test-time training. By enforcing the cycle consistencies across the forward and backward cyclic networks, test-time training is effectively conducted without accessing target labels. The designed structure intertwines four relevant training tasks to jointly optimize the model and bridge the training and test-time training stages.

- Extensive experiments and ablation studies are conducted on various real-world traffic datasets, and state-of-the-art performance has been achieved to verify the superiority and effectiveness of each component of the proposed methods.

1.4 Thesis Organization

This thesis is organized as follows:

- *Chapter 2*: This chapter presents a survey of spatial-temporal representation learning consisting of CNNs, GCNs, RNNs, and the Transformer, as well as the applications that apply these models to achieve urban traffic forecasting tasks.

- *Chapter 3*: This chapter proposes a bidirectional Spatial-temporal adaptive Transformer (Bi-STAT) for accurate traffic forecasting model, which aims to automatically control the computation resources assigned according to different complexity of spatial-temporal forecasting tasks.

- *Chapter 4*: This chapter presents RiskContra, a contrastive learning approach with multi-kernel networks, to forecast the Risk of traffic accidents.

- *Chapter 5*: This chapter designs a novel embedded 2D spectral learning framework for exploring spatial-temporal forecasting in the frequency domain.

- *Chapter 6*: This chapter proposes a novel test-time training framework for spatial-temporal forecasting, which aims to mitigate the distribution shifts widely spread in the time series data between the training and testing stage.

- *Chapter 7*: The final chapter provides a summary of the thesis, as well as the potential future work in the spatial-temporal forecasting domain.

Chapter 2

Literature Survey

2.1 Spatial-Temporal Representation Learning

Due to the limitations of the statistical methods in dealing with the complex and non-linear spatial and temporal correlations in real-world traffic datasets, deep learning has become the dominant approach to traffic forecasting tasks, which is attributed to its powerful capabilities to represent hierarchical patterns and discriminative features. Specifically, CNNs, GCNs, RNNs, and Transformers (Fig. 2.1) are representative deep networks that have been widely adopted by most of the existing spatial-temporal forecasting and traffic forecasting applications. The combination of these models from spatial and temporal correlation modeling serves as the building block for most existing spatial-temporal forecasting models. Thus, it is fundamentally important to understand how these deep networks work from a spatial-temporal modeling perspective.

2.1.1 Convolutional Neural Networks

CNNs have been one of the most popular models in the community of image processing and computer vision [26, 29, 53], which were originally designed for dealing with image datasets to extract inherent invariant local neighboring relationships. Different from the regular-shaped image datasets, traffic data with spatial and temporal dimensions can not be directly fed into the CNNs. Thus, the researchers convert the traffic network at different time steps into regular-shaped maps and divide them into multiple grids representing different regions. For example, [95, 118, 123]

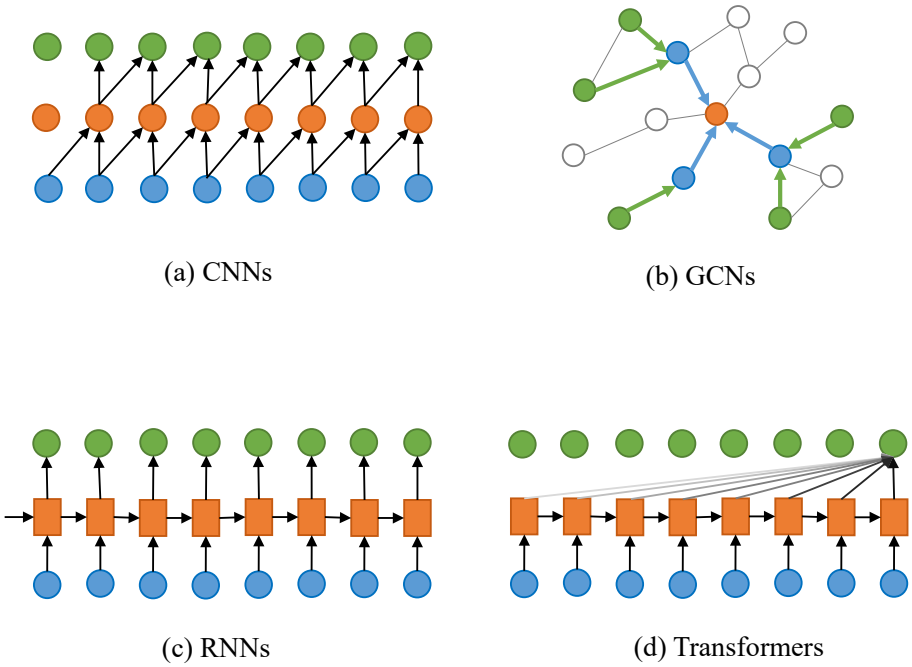


Figure 2.1 : The representative spatial-temporal representation learning networks.

partitioned a target city into $I \times J$ grids based on the longitude and latitude (I and J are the lengths of the grid map). In this way, CNNs can be applied to capture the spatial dependencies at a local level and learn representations that are shared across different regions. They have been proven successful in leveraging the inherent spatial structure of traffic data for accurate predictions and are widely used in the field of traffic forecasting.

CNNs mainly consist of three different kinds of neural network layers, which are the convolution layer, the pooling layer and the fully connected layer respectively. We review quickly about the main task of each layer as follows:

- The convolution layer plays a crucial part in extracting the dominant features from input data. Specifically, they employ a certain-sized kernel to the data and perform the convolution on it, which is the sum of total element-wise products.

- The pooling operation is applied to reduce the size of data, which can efficiently speed up the learning and prevent possible overfitting issues. The most commonly used versions of pooling are average pooling which computes the mean value of data in the given window and max pooling which outputs the max value instead.

- After the feature extraction through the convolutional layers, the outputs are fed into the fully connected layers, also known as dense layers, to obtain the final results. Notably, the data from the convolutional and pooling layers will be flattened before entering into the fully connected layers.

Except for learning the spatial dependencies in grid traffic data, recent works have demonstrated the effectiveness of applying specific convolutional techniques to the temporal modeling problems [27]. Different from the 2-dimensional spatial convolution, temporal convolution is only operated through the time dimension to carry out the non-linear transformation. Specifically, the causal convolution is commonly adopted to deal with time series datasets [4, 8, 71], in which convolutional filters are revised to adopt only the previous information with future steps masked for more reasonable forecasting intuition.

A great challenge will emerge when directly applying the standard convolutional networks to long-range time series datasets, which will cause a heavy computational burden. The reason can be attributed to the limited receptive field of CNNs with only the specific kernel size to be operated at one time, and this will hinder the CNNs from capturing long-term temporal relationships. A solution is to adopt the dilated convolutional layers in the CNN architectures to reduce the computation cost [4, 71]. By skipping values with a certain step while sliding over the input data, it can exponentially enlarge the receptive field with the layer depth increasing.

Given a 1-dimensional data temporal input $x \in R^T$ and a convolution filter

$f \in R^K$, the dilated causal convolution can be represented as:

$$x \star f(t) = \sum_{s=0}^{K-1} f(s)x(t - d \times s)\,, \tag{2.1}$$

where $\star$ is the convolution operator and d is the dilation factor to control the skipping steps.

2.1.2 Graph Convolutional Networks

Although CNNs have proven highly effective for grid-structured data, they face challenges in handling irregular graph-structured data, such as road networks, where nodes (entities) can have varying degrees and non-uniform connections. Thus, the application of CNNs to traffic forecasting is limited to modeling the correlations in the Euclidean space, which leads to unsatisfying forecasting performance without consideration of graph connections in the traffic network.

Aiming to extend the CNNs to the non-Euclidean space, GCNs provide a framework to extend convolutional operations to graph-structured data. They enable nodes in a graph to aggregate information from neighboring nodes, accommodating variations in connectivity and degree.

The core component of a GCN is the graph convolutional layer, which is defined as follows [51]:

$$f(H^{(l)}, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})\,, \tag{2.2}$$

where A is the adjacency matrix, $\hat{A} = A + I$ with I being the identity matrix and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$, $W^{(l)}$ is a weight matrix for the l-th neural network layer, $H^{(l)}$ is the hidden representation learned from the $l-$th layer of GCN, and $\sigma(\cdot)$ is a non-linear activation function such as the ReLU.

In this way, traffic forecasting can be conditioned on the traffic network and the convolution can learn the correlations among regions in the non-Eulidean space. Formally, a traffic network is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ is the

set of nodes representing the traffic locations, $|\mathcal{V}| = N$ is the number of sensors, $\mathcal{E}$ is the set of edges reflecting the correlations among different regions and $A \in \mathbb{R}^{N \times N}$ is the adjacent matrix of the graph to measure the proximity between nodes (i.e. spatial correlations).

The adjacency matrix of the graph is usually computed from the distances among the nodes in the traffic network by the thresholded Gaussian kernel as follows:

$$A_{ij} = \begin{cases} \exp(-\frac{\text{dist}(v_i, v_j)^2}{\delta^2}), & i \neq j \text{ and } \text{dist}(v_i, v_j) \leq \epsilon \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

where A_{ij} represents the edge weight between the node pair v_i and v_j, $\text{dist}(v_i, v_j)$ represents their distance in the traffic network. δ and ϵ are thresholds parameters.

Although GCNs are widely applied in traffic forecasting tasks, the graph structure is static, which violates the dynamic spatial correlation in time-varying traffic networks, Arguing that the graph structure in spatial-temporal forecasting should not be static, Thus, some more recent literature proposed the self-adaptive adjacency matrix, which does not rely on prior structural knowledge and is learned end-to-end in the forecasting model. Specifically, two randomly initialized node embedding matrices are multiplied to infer the spatial dependencies between each pair of nodes:

$$\hat{A}_{adapt} = \text{Softmax}(\text{ReLU}(E_1 {E_2}^T)\,. \tag{2.4}$$

where E_1 and E_2 respectively represent the node embedding to be learned in the model.

2.1.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are specifically proposed to model sequential data, such as the sequences of words in natural language processing (NLP) tasks, the audio data in speech recognition problems, and time series in forecasting

problems. All of these problems share a common property that they have temporal dependencies with time evolving. RNN-based models have the natural superiority to capture this kind of dynamic correlation over variable periods, which the traditional feed-forward neural networks cannot take into account.

Generally, the core component of an RNN cell is a central memory state that integrates the past information into a compact summary. Then, this memory state receives the newly observed input at each current time step to update its information recursively.

Different from CNNs which need to specify the explicit window to look back at neighboring time steps, RNNs process signals in a sequential fashion, and this significantly increases the receptive field of the model. Although having made a lot of achievements in the sequence learning domain, conventional RNNs have a notoriously limited capacity to be trained on long sequences on account of the gradient vanishing and exploding problem. Specifically, as the number of layers increases, the gradient becomes null in practice, which heavily hinders the back-propagation mechanism during the learning process of the neural network. This is why these standard RNNs can only perform well on the short-term sequences and fail to achieve good results on the long sequences which need to be memorized about the information at all the time steps.

LSTM

The emergence of long short-term memory recurrent neural networks (LSTMs) [88] sheds light on tackling the gradient vanishment challenge, which revises the architecture of the original recurrent neural network by employing three functional gates to selectively memorize the important relevant information and simultaneously abandon the useless information. To be specific, these gates are referred to as the forget gate G_f, the update gate G_u, and the output gate G_o respectively, according to their

different functionality. As names imply, the forget gate is used to control whether the information should be neglected or kept for further use. Concretely, if the value from the forget gate is close to 0, then the corresponding past information should be forgotten while a value of approximately 1 is a signal for important information to remain. Besides, the update gate is devised to decide whether to update the central memory state according to the new information. Note that the central memory is refreshed by activating both the forget and update gates. The output gate controls the final input to the next hidden cell considering all the given conditions.

The information from historical hidden states and present inputs are fed into the sigmoid function to obtain all of the gate values and into the tanh function to attain the new information for the update step. We formulate all these procedures in the following equations to formalize the definition of LSTM:

$$z_t = \tanh(W[h_{t-1}, x_t] + b)\,, \tag{2.5}$$

$$c_t = z_f \times c_{t-1} + z_u \times z_t\,, \tag{2.6}$$

$$h_t = z_o \times \tanh(c_t)\,, \tag{2.7}$$

$$z_f = \sigma(W_f[h_{t-1}, x_t] + b_f)\,, \tag{2.8}$$

$$z_u = \sigma(W_u[h_{t-1}, x_t] + b_u)\,, \tag{2.9}$$

$$z_o = \sigma(W_o[h_{t-1}, x_t] + b_o)\,, \tag{2.10}$$

where W_f, W_u, W_o W_c, b_u, b_r, b_o, and b_c are respectively the weights and bias for the forget gate, update gate, output gate, and memory cell.

GRU

The Gated Recurrent Unit (GRU) is an improved version of LSTMs for efficiently dealing with long-term memory sequence learning in recurrent neural networks. It was first proposed in [20] and referred to as a simplification of LSTMs to make up for its deficiency of high computational burden. GRU is widely applied to various

fields and has proven to be more robust and useful than other variants of RNNs. The major difference between the GRU and LSTM exists in the way they apply the gate, which is the key to success for the GRU to effectively capture long-range dependencies. Specifically, the GRU owns two gates compared to the three gates in LSTMs, which leads to the major differences between the two RNN variants, more powerful and effective for the LSTM and simpler and more computationally efficient for the GRU. The two gates are the relevance gate z_r and update gate z_u. Similarly to the gates in LSTM, the update gate decides for the c_t central memory about whether to be updated by the candidate memory state z and the z_r gate measures the relevance between the last and current memory state to compute the next candidate for memory state. Formally, GRU is defined in the following:

$$z_u = \sigma(W_u[c_{t-1}, x_t] + b_u)\,, \tag{2.11}$$

$$z_r = \sigma(W_r[c_{t-1}, x_t] + b_r)\,, \tag{2.12}$$

$$z_t = \tanh(W_c[z_r \times c_{t-1}, x_t] + b_c)\,, \tag{2.13}$$

$$h_t = c_t = z_u \times z_t + (1 - z_u) \times c_{t-1}\,, \tag{2.14}$$

where W_u, W_r, b_u, b_r, W_c, and b_c are respectively the weights and biases for the reset gate, update gate, and memory state.

2.1.4 Transformer

The development of attention mechanism [91] provides a promising way for modeling the long sequences, which was originally devised in the NLP domain to enrich the embedding of each word with the surrounding context in a sentence. The attention mechanism dynamically builds the relation according to the similarity between each word and all of the remaining words in the same sentence. Then it gathers the information from the most relevant context to extract better meaning from each word for the subsequent tasks.

The Transformer model is the successful employment of the self-attention mechanism as a non-local operation in an encoder-decoder architecture to process a sequence of data. It is widely applied to various domains including NLP, image classification, and video representation. The main reason for the Transformer's dominance and widespread popularity across various domains is its capability to attend to critical context information. In contrast to the RNN-based methods, the Transformer has a larger receptive field which enables it to access any distant part of historical information, which endows it with great capability to capture the complex and dynamic dependencies in the long-range sequences.

Conceptually, multi-head attention in the Transformer model is a key-value lookup based on a given query, which takes the following form:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_\text{h})W^O\,, \tag{2.15}$$

$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\,, \tag{2.16}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V\,, \tag{2.17}$$

where W^O, W_i^Q, W_i^K and W_i^V are the weight parameters, h is the number of head, and d_k is the dimension of key.

Due to the similarity of sequential language data and the time series data, the attention mechanism can be naturally applied to the time series forecasting tasks with significant improvement [24, 47, 57, 128]. By dynamically allocating weights to different time steps, the attention mechanism allows time steps in a sequence to be aware of its important historical context, which significantly promotes forecasting performance. Furthermore, by applying the attention mechanism to both the spatial and temporal dimensions respectively in traffic data, Transformer-based traffic forecasting models have been successfully developed [9, 12, 33, 62, 107, 127]. These methods provide a significant gain in forecasting performance over the other deep neural networks such as convolutional networks and recurrent networks-based

forecasters, which can account for the advantage of the attention mechanism to selectively attend more to important parts and capture more useful latent representations in the historical data for better forecasting performance.

2.2 Applications in Urban Traffic Forecasting

In the early stage, statistical models such as Historical Average (HA) [101], autoregressive integrated moving average (ARIMA) [102] and VAR [133] were proposed to perform the traffic forecasting task. However, these traditional methods struggle to capture the complex nonlinear spatial-temporal correlations hidden in the large-scale traffic streams.

Recently, the development of deep learning [53] provides a promising direction to solve this problem because of its capability to extract discriminative features and the ability to capture complicated spatial-temporal correlations [44, 75, 120]. Therefore, an assortment of deep neural networks are applied to the traffic forecasting problem, such as RNNs [88, 108], GCNs [16, 51], CNNs [53], attention and Transformers [91]. RNNs [88] have a natural superiority in modeling sequential dependencies and thus have been widely applied to traffic prediction problem [3, 60, 63, 73, 117, 119].

2.2.1 Recurrent Neural Networks

In particular, LSTM and GRU are two popular RNN variants that have been adopted to learn the temporal dependencies of the traffic streams, which are incorporated in an encoder-decoder architecture where spatial correlations are learned by GCNs, CNNs, or attention. For example, [119] utilizes the ConvLSTM structure to model long-term trends and short-term variations of the mobile traffic volumes. ST-MetaNet [73] employs GRU as a concrete example of RNNs in their method to encode the temporal dynamics of urban traffic. [118] designs different ConvLSTM modules for different regions to simulate the spatial heterogeneity of traffic

accidents. [131] adopts a multi-task learning scheme that employs traffic volume prediction as an auxiliary task for minute-level traffic accident forecasting. [95] designs a CNN-GRU module to capture the geographical spatial-temporal correlations and a GCN-GRU module to explore the semantic spatial-temporal correlations. [113] decomposes spatial demand into multiple bases and integrates them by a heterogeneous recurrent model to achieve joint prediction.

2.2.2 Convolutional Neural Networks

CNNs have also been recently applied to time-series prediction problems, such as temporal convolution networks. Compared with RNNs, CNNs can be easily combined with GCNs to jointly model the spatial and temporal patterns, which is more suitable for traffic forecasting. For example, GraphWaveNet [106] develops an adaptive dependency matrix to complement the potential dependency that may be lost in a fixed graph matrix. And they also employ the dilated causal convolution to capture long-range temporal sequences. STGCN [115] is the first study to apply convolutional structures to capture both spatial and temporal patterns. It is designed to be a multi-convolutional block architecture that employs the graph convolutional layers and convolutional sequence layers to model the spatial and temporal dependencies, respectively.

2.2.3 Graph Convolutional Networks

Traffic forecasting is not a simple time-series prediction problem, it also involves a complicated spatial traffic network. The traffic network is constructed by all the locations in the target city, which have irregular and elusive inter-dependencies. To effectively model the spatial correlations among roads, GCNs [51] have been introduced in traffic forecasting [2, 3, 17, 56, 60, 86, 105, 106, 111, 115]. For example, DCRNN [60] re-formulates the spatial dependency in traffic forecasting as a diffusion process on a directed graph, which is specifically achieved by the bidirectional

random walks on the graph.

To break the constraints of the pre-defined static graph, AGCRN [3] devises two adaptive modules and incorporates them into a GRU model to form a unified traffic forecasting model. Therefore, it can capture node-specific spatial and temporal correlations in the traffic streams. [23] aims to localize spatial-temporal graph models by sparsitying spatial graph adjacency matrices in the adaptive spatial-temporal GNN models to achieve full localization. It can effectively reduce the computation burden on large-scale spatial-temporal data and thus enable the distributed deployment of adaptive spatial-temporal GNNs. [35] proposes a spatial-temporal GNN model that fuses positional, topological, and temporal information into rich inductive node representations through a combination of gated Lipschitz embeddings with LSTMs, which proves more expressive than message-aggregation-based spatial-temporal GNNs. Instead of following the assumption that the spatial correlations among regions are static, [58] targets time-aware spatial dependency learning. They propose a spatially evolving graph convolution module to perform location representation learning by time-aware self-supervised learning with evolving constraints. [45] proposes a Meta-Graph Learning scheme for spatial-temporal graph learning, which is incorporated into a Graph Convolutional Recurrent Network to explicitly address the heterogeneity and non-stationarity problems in traffic data. [42] also targets the spatial-temporal heterogeneity caused by the time-varying traffic distributions. It proposes a Spatial-Temporal Self-Supervised Learning framework for traffic prediction, which is built based on temporal and spatial convolutions, with the attribute- and structure-level graph data augmentation designed for conducting two self-supervised auxiliary tasks. [97] designs a channel-wise CNN and multi-view GCN to capture both the local geographic and global semantic dependencies for traffic accident forecasting. [2] conducts multi-step passenger demand forecasting by stacking GCNs in a hierarchical way to extract spatial correlations

and employing a temporal attention mechanism to capture the dynamic influence of historical features. [31] proposes to model the spatial-temporal dependencies of bike flows by a sparse representation with graph regularization to preserve the common structure of inputs. [114] proposes a GCN-based architecture with a different self-learned adjacency matrix incorporated into each layer of the model, and the hidden spatial states are integrated into an RNN model to capture the dynamic spatial and temporal correlations.

2.2.4 Transformer

Transformer [91] has been widely used in various spatial and temporal applications. The core building block of the Transformer is the attention mechanism, which is able to dynamically learn the adaptive correlations between the input features and gather the auxiliary information from the most contributing ones for output features. Compared with RNNs, Transformer enjoys more efficient computation due to the parallelization-in-time mechanism, while RNNs require iterative computation along time steps. This way, Transformer can better handle the long sequence modeling than RNNs and avoid the gradient exploding and vanishing problems. Due to the prominent sequence modeling ability, Transformer has been applied to the traffic forecasting problem recently [9, 12, 33, 62, 107, 127]. These works either design task-specific attention modules similar to Transformer or directly utilize the Transformer architecture. Specifically, ASTGCN [33] employs a spatial-temporal attention mechanism to learn the dynamic spatial-temporal correlations of traffic data. [40] leverages the heterogeneous contextual knowledge and utilizes the attention mechanism to capture the temporal correlations for traffic accident forecasting. STTNs [107] proposes a spatial transformer to dynamically model directed spatial dependencies and a temporal transformer to capture the long-range temporal dependencies over time. To model the periodicity inherent in the time series,

Traffic Transformer [9] proposes a deep learning model to capture the continuity and periodicity in the time series, where the spatial dependencies are captured by the GCN model and Transformer is leveraged to model the temporal dependencies. GeoMAN [62] proposes a multi-level attention mechanism based on an encoder-decoder architecture. The encoder consists of two different spatial attentions to model the local and global relations, while the decoder consists of a temporal attention to model the dynamic temporal correlation between different time intervals. GMAN [127] proposes a graph multi-attention network for traffic prediction, which follows the encoder-encoder architecture in the transformer and applies it to both the spatial and temporal dimensions to capture the dynamical dependencies. To avoid the large accumulative errors brought by the autoregressive decoding of the canonical Transformer model, NAST [12] proposes a query generation block module between the encoder and decoder to generate queries for the spatial-temporal Transformer decoder. TrafFormer [46] integrates the spatial and temporal features into a Transformer model. It designs a spatial-temporal correlation matrix to allow each time step to interact with each other in just one step to capture the complex inter-dependencies in traffic data. PDformer [43] applies self-attention with two graph masking matrices designs to highlight the short-term and long-term dynamic spatial dependencies. Besides, a traffic delay-aware feature transformation module is proposed to solve the time delay during the propagation of traffic states between different locations. Besides capturing the spatial and temporal correlations, [126] proposes a Transformer model to capture the cross-dimensional correlations among different features in multivariate time series.

Chapter 3

Bidirectional Spatial-Temporal Adaptive Transformer for Urban Traffic Flow Forecasting

3.1 Introduction

Due to the complicated and dynamic spatial-temporal correlations, traffic forecasting is modeled as a spatial-temporal graph modeling problem [100, 106], where all sensors on the roads constitute a spatial-temporal graph to represent the evolving traffic conditions. In order to predict the future traffic condition of each node, recent studies on spatial-temporal graph modeling mainly integrate graph convolution networks (GCNs) [51] with different time-series models, i.e., recurrent neural networks (RNNs) [3,60], convolution neural networks (CNNs) [66,106,115,122,123], and Transformer [12,33,62,107,127]. Both the graph structure and the present traffic condition are incorporated for future condition prediction. While existing methods show the effectiveness of modeling the spatial-temporal dependency, two intrinsic properties of the traffic forecasting problem are overlooked, which hinders existing methods from becoming more efficient and accurate traffic condition predictors.

First, the complexities of diverse forecasting tasks are imbalanced with respect to both spaces (e.g. suburb vs. downtown) and times (e.g. rush hour vs. off peak). As shown in Figure 3.1, the traffic flow of road 1 (in downtown) exhibits higher values and more complex patterns compared with those of road 3 (in suburb). Meanwhile, for a specific road, the rush hour flow, oscillating frequently, is harder to predict than the off peak flow, which shows a stable and simple pattern. As a result, the imbalanced task complexity poses considerable difficulties for accurate

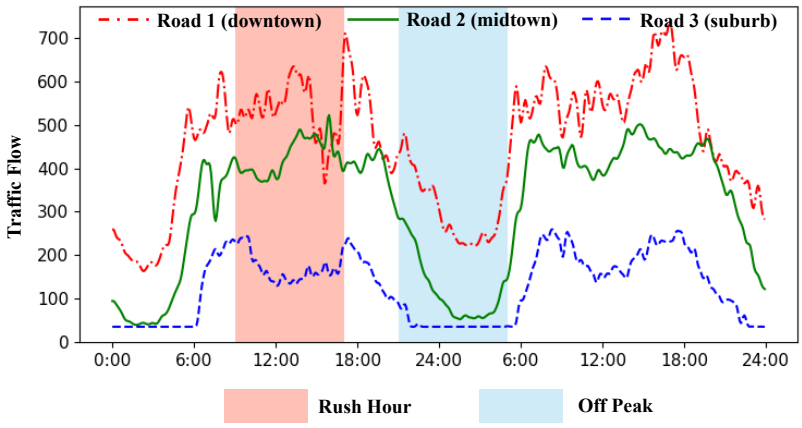


Figure 3.1 : The traffic forecasting tasks exhibit imbalanced complexities across varying spaces and times. (1) Downtown (e.g. road 1) shows higher values and more complex patterns than suburb (e.g. road 3). (2) The rush hour oscillates wildly while the off peak shows a stable and simple pattern.

traffic forecasting. However, existing methods allocate equal computation load to all tasks regardless of their spatial-temporal complexities. Consequently, important details are neglected for complex tasks due to limited computation while excessive computation loads are wasted for simple tasks. Thus, a dynamic and adaptive computation model is required to address the aforementioned issues.

Second, the recollection of past traffic conditions plays an important role in the prediction of future traffic conditions. Recent neuroscience studies [70, 81] find that remembering the past and predicting the future are intimately linked in a common brain network. This intuition is also verified and incorporated in various practical fields, such as Natural Language Processing (NLP) [21], and video representation [132]. However, in traffic forecasting, existing methods only focus on the future prediction task, ignoring the effect of past recollection. Since the traffic forecasting model is usually trained in one period and tested in another period, only considering the future prediction will cause the model to overfit the training period. This issue can be alleviated by the incorporation of the past recollection task, which will

prevent the model from aggressively fitting the future prediction and bring extra context information for consistent prediction.

Motivated by the two properties, this chapter proposes a Bidirectional Spatial-Temporal Adaptive Transformer (Bi-STAT) for accurate traffic forecasting. Specifically, Bi-STAT targets at the traffic flow forecasting problem, which exhibits the dynamic varying spatial-temporal patterns of the traffic flow information shown in Figure 3.1. Bi-STAT is designed to be an encoder-decoder architecture. Each encoder (decoder) maintains a spatial-adaptive Transformer to model the spatial dependency and a temporal-adaptive Transformer to capture the temporal pattern, which are then entangled to explore the spatial-temporal correlation. To address the first property, each Transformer is devised to dynamically process the traffic streams according to the varying task complexity. Specifically, we first utilize the recurrent inductive bias of RNNs to perform iterative computation with shared parameters. Then a novel Dynamic Halting Module (DHM) is devised to estimate the required computation for each task and terminate the computation when necessary. The recurrent design is parameter-efficient, while the DHM is flexible and efficient by parsimoniously assigning the essential computation load for each task. To cope with the second property, at training time, we employ two independent decoders to perform the *present→past recollection* task and the *present→future prediction* task, respectively. The recollection task provides extra context information and serves as a regularizer to prevent the prediction task from excessively fitting the future stream. It is notable that we remove the DHM module in our recollection decoder. The reason is that we count on this DHM in the future prediction decoder to dynamically allocate the computation resources for better learning the future traffic representation. By contrast, we only need the past recollection decoder to serve as a simple regularizer by learning the historical traffic representation. The different mission of the two decoders lead to the difference in architecture design, which al-

lows our model to focus more on the target future prediction task and achieve better performance gains.

3.2 Method

3.2.1 Problem Formulation

We target at the multi-step traffic flow forecasting problem, i.e. predicting the future traffic flow conditions of multiple time steps simultaneously.

Different from time-series prediction tasks, traffic forecasting is conditioned on the traffic network. Formally, a traffic network is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ is the set of nodes representing the traffic sensors, $|\mathcal{V}| = N$ is the number of sensors, $\mathcal{E}$ is the set of edges, and $A \in \mathbb{R}^{N \times N}$ is the adjacent matrix of the graph to measure the proximity between nodes (i.e. spatial correlations).

Based on the traffic network, we can define the traffic flow forecasting problem for all N sensors. At first, different from existing traffic forecasting models which only utilize the present traffic data to predict the future traffic conditions, we adopt both the historical traffic sequences $\mathcal{X}^H = (X_1, X_2, \ldots, X_H) \in \mathbb{R}^{H \times N \times d}$ and the present traffic sequences $\mathcal{X}^P = (X_{H+1}, X_{H+2}, \ldots, X_{H+P}) \in \mathbb{R}^{P \times N \times d}$. Here, H and P stand for the historical and present time steps respectively, d denotes the number of features in traffic conditions (e.g. traffic flow, traffic speed, etc.), which is 1 in our method to only represent traffic flow. Based on $\mathcal{X}^H$ and $\mathcal{X}^P$, our goal is to predict the conditions of future F time steps, denoted as $\mathcal{X}^F = (X_{H+P+1}, X_{H+P+2}, \ldots, X_{H+P+F}) \in \mathbb{R}^{F \times N \times d}$. To achieve this goal, we train a model f parameterized by θ to predict an approximation of $\mathcal{X}^F$ as follows:

$$\hat{\mathcal{X}}^F = f_\theta(\mathcal{X}^H, \mathcal{X}^P; \mathcal{G}) \,. \tag{3.1}$$

To be consistent with existing methods, at testing time, we only utilize $\mathcal{X}^P$ and $\mathcal{G}$ for prediction. The effect of $\mathcal{X}^H$ is to regularize the model f and provide extra

context information at training time.

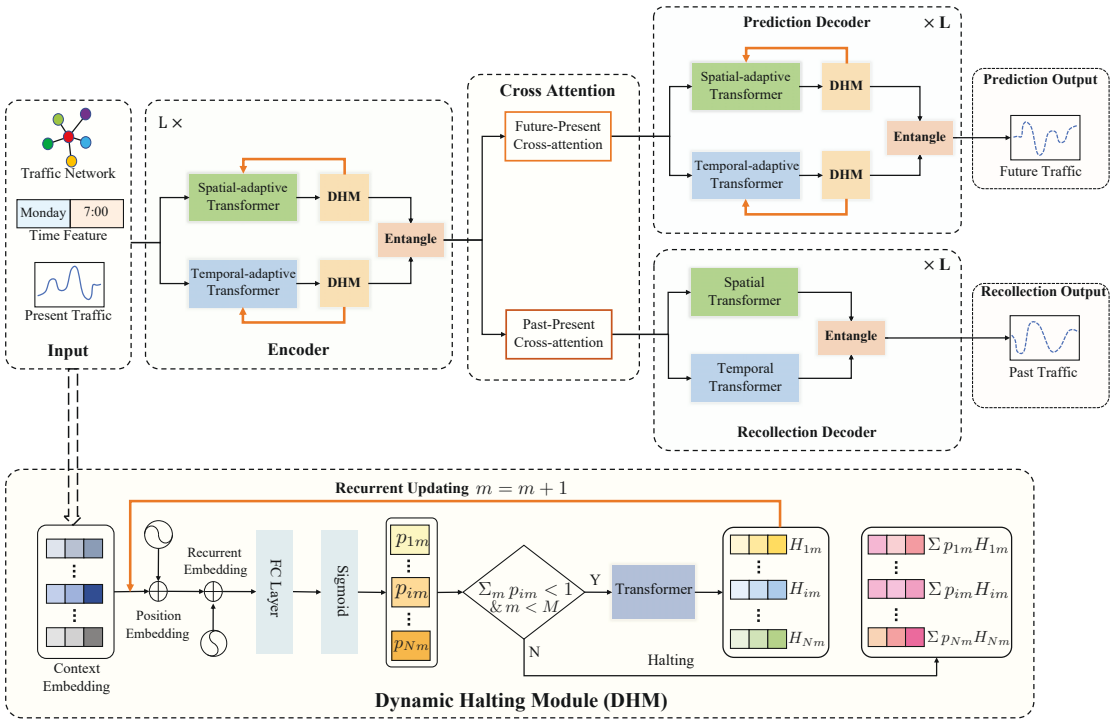


Figure 3.2 : The framework of the Bidirectional Spatial-Temporal Adaptive Transformer (Bi-STAT). Bi-STAT consists of an encoder, a prediction decoder, and a recollection decoder. In both the encoder and prediction decoder, we realize the task-adaptive computation (spatially and temporally) by the recurrent mechanism with a novel Dynamic Halting Module (DHM). The recollection decoder performs the past reconstruction task to provide extra context information and regularize the future prediction model.

3.2.2 Framework Overview

The overview of the Bidirectional Spatial-Temporal Adaptive Transformer (Bi-STAT) framework is shown in Figure 3.2.

There are three sources of inputs in our traffic forecasting problem: the traffic network (***spatial context***), the time feature (***temporal context***), and the present

traffic streams (***input feature***), which we will elaborate in the next subsection. After pre-processing and feature embedding, we input embeddings to the encoder. The encoder contains multiple layers with each layer maintaining a spatial-adaptive Transformer, a temporal-adaptive Transformer, and a Dynamic Halting Module (DHM) architecture. In the encoder, the task-adaptive computation is realized by the recurrent Transformer computation with the DHM to dynamically control the number of computation steps and integrate the weighted sum of the output from the Transformer. After that, the spatial and temporal features are entangled to capture the inter-dependencies among the traffic nodes at different times.

Bi-STAT contains two decoders. The prediction decoder shares the same architecture with the encoder and is used for future traffic condition prediction. The recollection decoder has a simpler structure without the DHM and serves as the regularizer to prevent the prediction decoder from aggressively fitting the future traffic conditions. Between the encoder and decoder, we design the cross-attention module, which models the direct relationship between each future (past) time step and all the present time steps. This design disconnects the links among different future (past) time steps, which has a potential effect of mitigating the notorious error propagation issue in the traffic forecasting problem.

3.2.3 Pre-processing and Input Embedding

In this subsection, we describe the three kinds of inputs mentioned in 3.2.2 in detail. The utilization of the spatial and temporal context information enables our model to be aware of the spatial-temporal surroundings of the prediction task. Moreover, these contexts offer important clues for the spatial-temporal complexity of the prediction task, which can be captured by our model to dynamically assign the computation load based on task complexity.

Spatial Context

We first construct the traffic network $\mathcal{G}$. Specifically, we calculate all the pairwise Euclidean distances among all the traffic sensors in the traffic network. These distances then form the adjacent matrix A of the graph. To convert the graph nodes (i.e. traffic sensors) to feature embeddings, we adopt the widely-used *node2vec* [30] approach. With *node2vec*, the graph nodes are represented by the spatial embedding SE. Since SE is static, we feed it into a two-layer fully-connected neural network to obtain the learnable spatial embeddings $SE \in \mathbb{R}^{N \times D}$.

Temporal Context

To build the temporal context, we employ two important temporal signals: *hour in a day* and *day in a week*. These two signals offer both short-term and long-term context information for the traffic prediction task. For example, the rush hour and weekend are strong signals to influence the short-term and long-term temporal evolution. In particular, a day is divided into T_D time steps and a week is divided into 7 days. They are then encoded as one-hot vectors of dimension $\mathbb{R}^{T_D}$ and $\mathbb{R}^7$, respectively, which are concatenated as $\mathbb{R}^{T_D+7}$ and fed into a two-layer fully-connected neural network. For different traffic sequences $\mathcal{X}^H, \mathcal{X}^P$, and $\mathcal{X}^F$, we get the corresponding temporal embedding as $TE^H \in \mathbb{R}^{H \times D}, TE^P \in \mathbb{R}^{P \times D}$, and $TE^F \in \mathbb{R}^{F \times D}$ respectively.

Input Feature

It is the original present traffic sequence $\mathcal{X}^P \in \mathbb{R}^{P \times N \times d}$. We also utilize a two-layer fully-connected neural network to align the dimension with the spatial and temporal embeddings SE and TE^P. After that, we get $\mathcal{X}_1^P \in \mathbb{R}^{P \times N \times D}$.

With all the spatial embedding SE, temporal embedding TE^P, and input feature

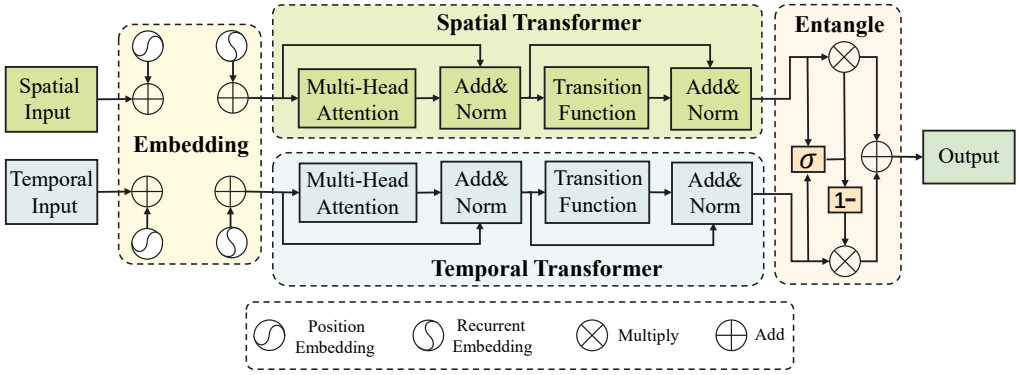


Figure 3.3 : The Spatial-Temporal Adaptive Transformer Architecture

$\mathcal{X}_1^P$ at hand, we can obtain the final input embeddings $E \in \mathbb{R}^{P \times N \times 2D}$ as follows:

$$E[i,j,:] = \text{Concat}(\mathcal{X}_1^P[i,j,:], SE[j,:] + TE^P[i,:]), \tag{3.2}$$

3.2.4 Spatial-Temporal Adaptive Transformer

To effectively capture the spatial-temporal correlation and patterns from the traffic sequences, we propose a Spatial-Temporal Adaptive Transformer (STAT) as shown in Figure 3.3. Specifically, we devise a spatial Transformer and a temporal Transformer to process the spatial input and temporal input, respectively. Before the spatial (temporal) input are fed into the spatial (temporal) Transformer, the position embedding and recurrent embedding are incorporated to provide the context information (e.g. sensor identity, time step, recurrent step). Specifically, the Transformer is composed of the Multi-Head Attention, layer Normalization with residual design and Transition Function. After the spatial Transformer and temporal Transformer processing, we further design an Entangle module to model the inter-correlations between the spatial and temporal sequences.

Before describing the Transformer details, we first discuss the design choice for dynamic computation. A straightforward design for dynamic computation is to first expand multiple sequential Transformer layers and then dynamically control the number of layers according to the task complexity. However, this design has two

disadvantages. First, directly expanding the Transformer layers will linearly increase the parameter number. If a large computation load is required, the Transformer will be inefficient in terms of parameter number and become unaffordable. Second, with dynamic computation, the deeper layers will receive less training data than the shallower layers, being less effective. To address these issues, we propose a different design in this chapter. Specifically, we utilize the *recurrent inductive bias* of the RNNs, which performs recurrent Transformer updating with shared parameters. For dynamic computation, we devise a Dynamic Halting Module (DHM) to control the number of recurrent updating steps (which will be described in the next subsection). This design enjoys the efficiency of the recurrent parameter-sharing mechanism and the flexibility of the dynamic computation by the DHM.

In the following, we first describe how to adapt the existing position embedding of Transformer to handle the recurrent Transformer updating. Then, we describe the architecture of Spatial Transformer, Temporal Transformer, and Entangle module, respectively.

Position Embedding and Recurrent Embedding

In the original design of Transformer, position embedding is devised to contain the relative position information of the sequence. Formally, the position embedding is computed as follows:

$$PE(p, 2i) = \sin(p/10000^{2i/D}) \,, \tag{3.3}$$

$$PE(p, 2i+1) = \cos(p/10000^{2i/D}) \,, \tag{3.4}$$

where p is the position index and i is the dimension of the embedding. In order to incorporate the recurrent step information into the Transformer, we devise a two-dimensional (position, step) coordinate embedding. For the positions $1 \leq p \leq T$

and recurrent steps $1 \leq m \leq M$, the coordinate embedding is computed as follows:

$$PE^m_{p,2i} = \sin(p/10000^{2i/D}) + \sin(m/10000^{2i/D})\,, \tag{3.5}$$

$$PE^m_{p,2i+1} = \cos(p/10000^{2i/D}) + \cos(m/10000^{2i/D})\,. \tag{3.6}$$

In Eq. 3.5 and Eq. 3.6, the first sin / cos calculates the position embedding, while the second sin / cos computes the recurrent embedding indicating the recurrent computation step.

Temporal-adaptive Transformer

Attention is the core component of Transformer. In this chapter, we use the scaled dot-product attention. Formally, the attention function combines queries Q, keys K and values V as follows:

$$\text{Attention}(Q, K, V) = \text{Softmax}\big(\frac{QK^T}{\sqrt{D}}\big)V\,, \tag{3.7}$$

where D is the number of columns of Q, K and V. A widely-used variant of attention in Transformer is the multi-head attention with k heads, i.e.

$$\text{MultiHeadAttention}(H^m) = \text{Concat}(\text{head}_1, \ldots, \text{head}_K)W^O \tag{3.8}$$

$$\text{where head}_i = \text{Attention}(H^m W^Q_i, H^m W^K_i, H^m W^V_i)\,. \tag{3.9}$$

Here, H^m stands for current state features with m denoting the recurrent computation step, $W^Q_i \in \mathbb{R}^{D\times D/k}, W^K_i \in \mathbb{R}^{D\times D/k}$, and $W^V_i \in \mathbb{R}^{D\times D/k}$.*

To include the recurrent inductive bias of RNNs into Transformer, we perform recurrent updating for the Transformer state H^m. Specifically, at recurrent step m, we update the state H^m as follows:

$$H^m = \text{LayerNorm}(A^m + \text{Transition}(A^m)) \tag{3.10}$$

$$\text{where } A^m = \text{LayerNorm}((H^{m-1} + PE^m) + \text{MultiHeadAttention}(H^{m-1} + PE^m))\,, \tag{3.11}$$

*The input layer weight dimensions are $\mathbb{R}^{2D\times D/k}$ since $E \in \mathbb{R}^{P\times N\times 2D}$.

where LayerNorm(·) [1] is the layer normalization adopted to accelerate the training of model, Transition(·) is the transition function that consists of two affine transformations with a ReLU activation between them.

Spatial-adaptive Transformer

The spatial-adaptive Transformer owns a similar structure as the temporal-adaptive Transformer, but has two differences in order to capture the spatial dependencies among traffic sensors.

First, the position embedding is different. In temporal-adaptive Transformer, the position embedding represents the time steps of the traffic sequences so that p ranges from 1 to T, $1 \leq p \leq T$ (e.g. $T = P$ for the present sequence $\mathcal{X}^P$). For spatial-adaptive transformer, we utilize position embedding to encode the sensor identities, i.e. $1 \leq p \leq N$.

Second, the effective self-attention dimension in Transformer is different. For temporal-adaptive Transformer, we reshape $\mathcal{X}^P$ to $\mathbb{R}^{N \times P \times D}$ where N is the batch dimension, P is the self-attention dimension for the temporal correlation. While for spatial-adaptive Transformer, $\mathcal{X}^P$ is reshaped to $\mathbb{R}^{P \times N \times D}$, where P is the batch dimension and N is the self-attention dimension for the spatial correlation.

Entangle Module

The spatial-adaptive Transformer and the temporal-adaptive Transformer respectively capture the spatial dependencies and temporal patterns in their own models. However, due to the complexity of the traffic forecasting problem, the spatial dependencies and the temporal patterns are usually entangled to exhibit complicated spatial-temporal correlations. For example, the rush hour of a downtown road will show the common patterns of both the downtown area spatially and the rush hour temporally.

Let H_S and H_T denote the features after the spatial Transformer and temporal Transformer, respectively. We first utilize a linear model with sigmoid activation to obtain the entangle gate g as follows:

$$g = \sigma(H_S \mathbf{W}_S + H_T \mathbf{W}_T + \mathbf{b}_g)\,, \tag{3.12}$$

where $\mathbf{W}_S \in \mathbb{R}^{D\times D}, \mathbf{W}_T \in \mathbb{R}^{D\times D}$ and $\mathbf{b} \in \mathbb{R}^D$ are learnable parameters, and $\sigma(\cdot)$ is the sigmoid activation. Then, the spatial features and temporal features are combined as

$$H = g \odot H_S + (1 - g) \odot H_T\,, \tag{3.13}$$

where $\odot$ denotes the element-wise product.

3.2.5 Dynamic Halting Module (DHM)

As described above, we utilize the recurrent updating for Transformer layers to realize the parameter-sharing recurrent computation. The recurrent mechanism increases the computation load to solve difficult and complex prediction tasks while it also keeps the parameter number unchanged, thus being parameter-efficient. However, the disadvantage is that the large computation load will be a waste for simple prediction task. To solve this issue, we propose a Dynamic Halting Module (DHM) to parsimoniously assign the necessary computation load according to the task complexity.

To be aware of the spatial-temporal task complexity, the initial input of DHM requires to contain the spatial context, the temporal context and the input feature, which coincides with the input embeddings E obtained in section 3.2.3. Moreover, since the DHM is applied to control the recurrent computation of Transformer, we should also include the position embedding and recurrent embedding to inform the DHM the current status of the recurrent computation. Through this analysis, we find that the Transformer state at a specific recurrent step, H^m, contains all the

aforementioned context information to evaluate the spatial-temporal task complexity. Thus we directly utilize H^m for the DHM.

Note that spatial and temporal Transformer own different DHMs. Here, we take the DHM attached to temporal Transformer as an example. Specifically, at each recurrent step m, given H^m, we first calculate the halting probability p^m as follows:

$$p^m = \sigma(H^m \mathbf{W}_h + b_h)\,, \tag{3.14}$$

where $\mathbf{W}_h \in \mathbb{R}^D, b_h \in \mathbb{R}^1$, and $p^m \in \mathbb{R}^P$ indicates the halting unit for all the P present sequences. Let p_i^m stand for the i-th sequence in particular. To determine if the computation step for the i-th sequence should be stopped to save the resource, we accumulate the halting probability over the m' recurrent steps as $\sum_{m=1}^{m'} p_i^m$. The stopping criteria is defined as follows:

$$\begin{cases} \text{stop for sequence } i, & \text{if } m = M \text{ or } \sum\limits_{m=1}^{m'} p_i^m >= 1-\epsilon \\ \text{continue}, & \text{otherwise} \end{cases} \tag{3.15}$$

where M is the maximum number of recurrent steps used to rigidly limit the computation costs, ϵ is a small constant to allow halting after a single recurrent step if $p_i^1 >= 1-\epsilon$.

With Eq. 3.15, we can define the real computation steps for each sequence i and the remainder value $R(i)$ as follows:

$$N(i) = \min\{M, \min\{m' : \sum_{m=1}^{m'} p_i^m \geq 1-\epsilon\}\}\,, \tag{3.16}$$

$$R(i) = 1 - \sum_{i=1}^{N(i)-1} p_i^m\,. \tag{3.17}$$

From Eq. 3.16, we see that the expected recurrent steps will be probably smaller than M.

To make full use of the halting probability, we utilize the weighted state features

to represent H^m

$$H_T[i] = \sum_{i=1}^{N(i)} p_i^m H^m[i] \,. \tag{3.18}$$

According to this weighted state representation, the halting probability not only controls the dynamic computation, but also re-weights the feature learning procedure.

3.2.6 Cross Attention Module

In sequence prediction tasks, a common strategy is to chronologically predict the sequence step by step and employ the prediction result from the previous step as the input of prediction for the next step. This manner, however, will cause the notorious error propagation effect between different prediction steps in the long term prediction, which has been observed in previous traffic forecasting work [3, 91, 107, 127]. To address this issue, we propose a Future-Present Cross-attention module between the encoder and the prediction decoder, as well as a Past-Present Cross-attention module between the encoder and the recollection decoder, respectively. The cross attention module directly models the relationship between each future (past) time step and all the present time steps. Due to space constraint, we only take the future-present module as an example and the formulation for the past-present module is similar. Formally, given the present time steps P_i ($P_i = 1, \ldots, P$), the future time steps F_i ($F_i = 1, \ldots, F$) and a sensor s_i, the spatial-temporal context for step P_i and step F_i is $SE[s_i, :]+TE^P[P_i, :]$ and $SE[s_i, :]+TE^F[F_i, :]$, respectively. We then model the correlation between future step F_i and present step P_i as

$$\begin{aligned} E_P &= SE[s_i, :] + TE^P[P_i, :] \\ E_F &= SE[s_i, :] + TE^F[F_i, :] \\ \lambda^{s_i}_{F_i,P_i} &= \frac{\langle E_F, E_P \rangle}{\sqrt{D}} \\ \lambda^{s_i}_{F_i,P_i} &= \frac{\exp(\lambda^{s_i}_{F_i,P_i})}{\sum_{P_j=1}^{P} \exp(\lambda^{s_i}_{F_j,P_i})} \,, \end{aligned} \tag{3.19}$$

where f is a fully-connected layer followed by the ReLU activation. With the correlation, we can recalculate the input to the decoder $H_D \in \mathbb{R}^{F \times N \times D}$ as

$$H_D[F_i, s_i, :] = \sum_{P_i=1}^{P} \lambda_{F_i,P_i}^{s_i} H[P_i, s_i, :]\,. \tag{3.20}$$

3.2.7 Decoders and Loss Function

Prediction Decoder

As shown in Figure 3.2, the prediction decoder shares the same structure as the encoder, with details described in section 3.2.4. The input to the prediction decoder is the encoded future representation from the Future-Present Cross-attention Module, where the correlations among the future time steps are isolated and each future time step independently correlates to all the present time steps. This way, the error propagation issue is alleviated.

Recollection Decoder

Previous traffic forecasting methods only employ the prediction decoder at training time. Due to the special property of the traffic forecasting problem, we can only utilize data from the previous and present duration for model training and receive data from the non-overlapping future duration for testing. This training and testing inconsistency makes the prediction decoder easily overfit the training duration and hard to generalize. On the other hand, according to recent neuroscience studies [70, 81], remembering the past plays an important role in the prediction of the future.

According to above analysis, we propose a recollection decoder to endue our model the ability to simultaneously predict the future and reconstruct the past. For the architecture design, a straightforward design is to directly copy the prediction decoder. However, this design has two potential issues. First, while the aim of the

recollection decoder is to alleviate the overfitting issue of the prediction decoder, directly adopting the design of the complex predictor decoder will increase the overall model parameters and further increase the overfitting. Second, if the recollection decoder is as powerful as the prediction decoder, then the model will try to perform the prediction and reconstruction task with the same efforts. This potential competition will interfere the future prediction task. Taking these into consideration, we devise the recollection decoder to be a simple structure: spatial Transformer and temporal Transformer without the DHM.

Loss Function

At training time, the overall losses include the historical reconstruction loss $\mathcal{L}_H$, the future prediction loss $\mathcal{L}_F$ and the penalty loss $\mathcal{L}_{DHM}$ from the DHM.

For $\mathcal{L}_H$ and $\mathcal{L}_F$, we utilize the mean absolute error (MAE) between the ground truth and the prediction (reconstruction) as follows:

$$\mathcal{L}_H = \frac{1}{H}\sum_{t=1}^{H} |\mathcal{X}^H - \hat{\mathcal{X}}^H| \,, \tag{3.21}$$

$$\mathcal{L}_F = \frac{1}{F}\sum_{t=1}^{F} |\mathcal{X}^F - \hat{\mathcal{X}}^F| \,, \tag{3.22}$$

where $\hat{\mathcal{X}}^H$ and $\hat{\mathcal{X}}^F$ denote the estimated reconstruction and the prediction made by our model respectively.

For $\mathcal{L}_{DHM}$, it is utilized to balance the required computation load and the model efficiency

$$\mathcal{L}_{DHM} = N(i) + R(i) \,. \tag{3.23}$$

Since $N(i)$ is an integer value ranging from 1 to M, it is not differentiable. In practice we treat $N(i)$ as constant and minimize $R(i)$ instead.

Table 3.1 : Dataset Statistics.

Dataset	# of sensors	Time duration
PeMSD3	358	09/01/2018 - 11/30/2018
PeMSD4	307	01/01/2018 - 02/28/2018
PeMSD7	883	05/01/2017 - 08/31/2017
PeMSD8	170	07/01/2016 - 08/31/2016

Finally, we can combine all the losses as

$$\mathcal{L} = \mathcal{L}_F + \alpha\mathcal{L}_H + \beta\mathcal{L}_{DHM}\,, \tag{3.24}$$

where α, β are two regularization terms to control the effect of the recollection decoder and the DHM.

3.3 Experiments

3.3.1 Datasets and Evaluation Metrics

Dataset Details. To demonstrate the effectiveness of our method, we conduct experiments on four large-scale real-world traffic forecasting datasets: PeMSD3, PeMSD4, PeMSD7, and PeMSD8. These datasets were collected by The California Department of Transportation (Caltrans) through its freeway Performance Measure System (PeMS) [11]. The detailed statistics of the four datasets are shown in Table 3.1. Due to their scale, diversity, and rich traffic conditions, these datasets are the most widely-used benchmarks for traffic forecasting problem [3,17,33,56,86,115].

Pre-processing and Data Splitting. Following recent methods [3, 17, 106, 127], all the four datasets are aggregated into 5-minute windows to generate 12 data points (time steps) per hour, and consequently 288 data points per day. Z-score normalization is applied to the traffic data for more stable training. To make fair

comparisons with existing methods, we chronologically allocate 70% of the data for training set, 10% for validation set, and 20% for testing set. Note that the training, validation and testing splits have no overlap in time duration.

Evaluation Metrics. We employ three commonly-used metrics [3, 17, 127]: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) to measure the performance of the prediction models.

3.3.2 Baselines

We compare the proposed Bi-STAT with the following baselines:

- DCRNN [60], which incorporates the diffusion convolution into the GRU module for multi-step traffic forecasting.
- STGCN [115], which employs the graph convolutional layers and convolutional sequence layers.
- ASTGCN [33], which employs the attention mechanism to capture the spatio-temporal dynamic correlations.
- GraphWaveNet [106], which is constructed by the graph convolution layer (GCN) and the gated temporal convolution layer (Gated TCN).
- STSGCN [86], which captures the localized spatial and temporal correlations individually.
- STFGCN [56], which constructs spatial and temporal graphs and combines them with a spatial-temporal fusion model.
- AGCRN [3], which designs two adaptive modules and incorporates them to a unified GRU model.

- Z-GCNET [17], which is a time-aware zigzag persistence based graph convolutional networks.
- GMAN [127], which adopts an encoder-decoder structure to include multiple spatial-temporal attention blocks.
- STTN [107], which incorporates spatial-temporal dependencies into the block of spatial and temporal transformer to achieve more accurate traffic forecasting.
- Traffic Transformer [9], which designs an encode-decoder architecture to consistently model the spatial and temporal dependencies in the traffic time-series by GCN and transformer respectively.

3.3.3 Experimental Setups

As described in Section 3.3.1, we pre-process the datasets to generate 12 data points (time steps) per hour. At training time, we utilize the historical one-hour and present one-hour traffic sequences as input and predict the future one-hour traffic sequences. To be consistent with previous methods, at testing time, only the present one-hour sequences are utilized for prediction. Specifically, we set all time steps to 12, i.e. $H = 12, P = 12$, and $F = 12$.

In our framework, the recollection decoder is mainly used for regularization and providing auxiliary information, so its structure is relatively simple compared to the encoder and the prediction decoder. In particular, the structure parameters involve the number of Transformer layers L, the number of heads K in multi-head attention, and the Transformer dimension D. For the recollection decoder, we set $L = 1, K = 3, D = 8$ on all the four datasets. For the encoder and the prediction decoder, we set $L = 2, K = 3, D = 8$ on PeMSD4 and PeMSD8, and $L = 1, K = 3, D = 8$ on PeMSD3 and PeMSD7. Moreover, we set a small loss weight α for the recollection

decoder loss, i.e. $\alpha = 0.001$ for PeMSD3 and PeMSD8, $\alpha = 0.01$ for PeMSD4 and PeMSD7.

For the Dynamic Halting Module (DHM), we have several parameters to adaptively adjust the computation steps, i.e. maximum recurrent steps M, the stopping threshold ϵ, the penalty weight β for DHM loss. We empirically set these parameters according to the validation set to balance the computation load and the accuracy. In particular, $M = 6$ in all experiments (unless stated otherwise), $\epsilon = 0.001$ for PeMSD3 and 0.0001 for other three datasets, $\beta = 0.001$ for all datasets. Note that our method does not rely on exhaustively tuning these parameters. Instead, they are used to obtain a desired computation-accuracy balance. We will demonstrate this in the following.

Our model is trained by the Adam optimizer with a learning rate of 0.001 for 100 epochs. The best model checkpoint is selected by the performance on the validation set.

Table 3.2 : Comparison with the State-of-the-Art methods on four datasets regarding three metrics (the smaller the better)

Model	Backbone	PeMSD3			PeMSD4			PeMSD7			PeMSD8		
		MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
STGCN [115]	GCN	17.49	30.12	17.15	22.70	35.55	14.59	25.38	38.78	11.08	18.02	27.83	11.40
ASTGCN [33]	GCN	17.69	29.66	19.40	22.93	35.22	16.56	28.05	42.57	13.92	18.61	28.16	13.08
STSGCN [86]	GCN	17.48	29.21	16.78	21.19	33.65	13.90	24.26	39.03	10.21	17.13	26.80	10.96
STFGNN [56]	GCN	16.77	28.34	16.30	19.83	31.88	13.02	22.07	35.80	9.21	16.64	26.22	10.60
Z-GCNETs [17]	GCN	-	-	-	19.50	31.61	12.78	-	-	-	15.76	25.11	10.01
GraphWaveNet [106]	GCN	19.85	32.94	19.31	25.45	39.7	17.29	26.85	42.78	12.12	19.13	31.05	12.68
DCRNN [60]	RNN	18.18	30.31	18.91	24.70	38.12	17.12	25.30	38.58	11.66	17.86	27.83	11.45
AGCRN [3]	RNN	16.75	28.60	16.23	19.83	32.26	12.97	21.10	34.99	8.93	15.95	25.22	10.09
GMAN [127]	Transformer	16.49	26.48	17.13	19.36	31.06	13.55	21.48	34.55	9.01	14.51	23.68	9.45
STTN [107]	Transformer	16.11	27.87	16.19	19.32	30.79	13.15	21.05	33.77	8.94	15.28	24.25	9.98
Traffic-Transformer [9]	Transformer	16.39	27.87	15.84	19.16	30.57	13.70	23.90	36.85	10.90	15.37	24.21	10.09
Bi-STAT (Ours)	Transformer	**15.30**	**25.80**	**15.46**	**18.53**	**29.96**	**12.37**	**20.28**	**33.24**	**8.50**	**13.58**	**23.08**	**9.21**

3.3.4 Comparison with the State-of-the-Art Methods

For a specific time step, the prediction can be generated from any previous one-hour training sequences with an interval (a.k.a horizon) ranging from 1 to 12 steps. **The small interval (horizon) is nearer to the training sequences, thus leading to a more accurate prediction.** To be consistent with other methods, we average the performance of all 12 intervals (horizons).

The overall comparison with nine state-of-the-art methods are shown in Table 3.2. It is obvious that the proposed Bi-STAT outperforms all other baselines with a large margin on all four datasets and three metrics. Among all the baselines, AGCRN [3] and GMAN [127] perform relatively well. To be specific, AGCRN designs two adaptive modules to learn the dynamic node-specific correlations and utilizes RNNs for sequence prediction, while GMAN employs several spatial-temporal attention blocks similar to Transformer. Our Bi-STAT beats both AGCRN and GMAN with a large margin, corroborating the effectiveness against both RNNs-based and Transformer-based strong competitors.

Table 3.3 : The effectiveness of the proposed Dynamic Halting Module (DHM) and the recollection decoder regarding various horizons. Horizon stands for the interval between training and testing sequences, e.g. horizon=3 means 15 minutes.

Model	Horizon=3 (15min)			Horizon=6 (30min)			Horizon=12 (60min)			Average		
	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
Transformer	18.44	29.67	12.84	19.22	30.94	13.40	21.08	33.54	14.89	19.36	31.06	13.55
Transformer+Recollect	18.19	29.34	12.31	18.89	30.51	12.76	20.40	32.71	13.87	18.99	30.58	12.93
Transformer+DHM	17.87	28.96	12.19	18.55	30.08	12.67	19.91	31.95	13.89	18.62	30.10	12.78
Bi-STAT (Ours)	**17.81**	**28.82**	**11.80**	**18.46**	**29.93**	**12.30**	**19.79**	**31.83**	**13.39**	**18.53**	**29.96**	**12.37**

3.3.5 Ablation Study

Effectiveness of the designed model components

We first investigate the effectiveness of the proposed Dynamic Halting Module (DHM) and the recollection decoder. To begin with, a simple baseline is constructed by removing both the DHM and the recollection decoder from our Bi-STAT model. We call this baseline model "***Transformer***" since only naive Transformers are utilized here. Then we add the DHM into the baseline model and call it "***Transformer+DHM***". Another variant model is the baseline model combined with the recollection decoder, called "***Transformer+Recollect***". Our model "**Bi-STAT**" incorporates all the proposed components.

We conduct experiments on the PeMSD4 dataset and the results are shown in Table 3.3. We can see that after introducing the recollection decoder (i.e. *Transformer+Recollect*), the performance is improved for all horizons and the average, showing the effectiveness of the recollection decoder. This also verifies the regularization effect of the recollection decoder to encourage the prediction model for better generalization. Besides, incorporating the DHM into the Transformer (i.e. *Transformer+DHM*) leads to large performance gains for all horizons and the average. The result demonstrates the effectiveness of DHM which assigns adaptive computation loads according to task complexities. Our complete model, combining both the recollection decoder and DHM, further improves the performance against each single model, showing the complementary effect of the two devised components.

In-depth Analysis of the Dynamic Halting Module

We further perform an in-depth analysis into the Dynamic Halting Module (DHM). At first, we study whether the performance improvements come from the increased computation or from the adaptive computation mechanism. To verify this, we construct a baseline with fixed recurrent computation steps, called

Table 3.4 : The performance and efficiency comparison between DHM and the fixed recurrent computation.

# Layers	Model	MAE	RMSE	MAPE (%)	$Steps_{\mathbf{S}}$	$Steps_{\mathbf{T}}$
$L = 1$	*Transformer*	19.48	31.01	13.48	1.0	1.0
	Transformer+Recurrent	19.17	30.77	12.99	6.0	6.0
	Bi-STAT (Ours)	**18.62**	**30.13**	**12.82**	1.1	4.1
$L = 2$	*Transformer*	19.36	31.06	13.55	1.0	1.0
	Transformer+Recurrent	19.04	30.64	12.78	6.0	6.0
	Bi-STAT (Ours)	**18.53**	**29.96**	**12.37**	2.4	2.8

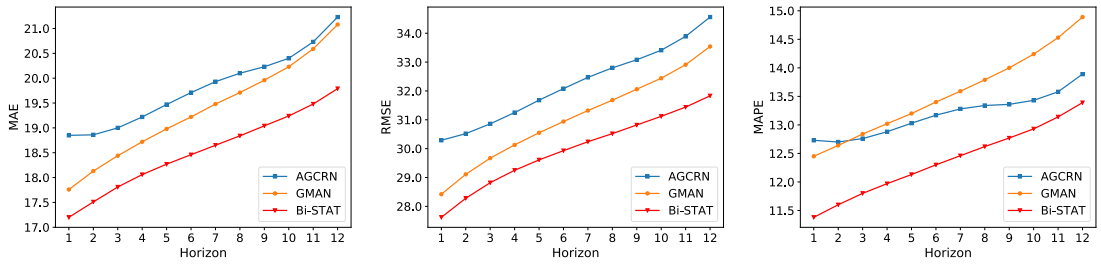
"***Transformer+Recurrent***". In contrast, our model "*Bi-STAT*" adopts the DHM mechanism to adaptively stop the computation, resulting in fewer computation steps. Here, we calculate the average computation steps for the spatial Transformer and temporal Transformer, denoted by $Steps_{\mathbf{S}}$ and $Steps_{\mathbf{T}}$. Also, we take the number of Transformer layers into account, i.e. $L = 1$ and $L = 2$. The results on the PeMSD4 dataset are shown in Table 3.4. With 6 computation steps, *Transformer+Recurrent* only slightly increases the performance. Our *Bi-STAT* employs fewer computation steps both spatially and temporally, but achieves much better performance. This demonstrates that our method does not rely on the increased computation to boost the performance. Instead, the adaptive computation mechanism is the key to our success.

From Table 3.4, we also observe that the average spatial steps and temporal steps are different. So we insert DHM into different positions of our model, i.e. spatial Transformer of the encoder ($Encoder_{\mathbf{S}}$), temporal Transformer of the encoder ($Encoder_{\mathbf{T}}$), and the prediction decoder ($Decoder$). The results on the PeMSD4 dataset are shown in Table 3.5. We can find that both the spatial and temporal Transformer of the encoder are benefited from the DHM, verifying the effectiveness

Table 3.5 : The effect of the DHM inserted into different positions of our model, e.g. spatial Transformer of the encoder ($Encoder_{\mathbf{S}}$).

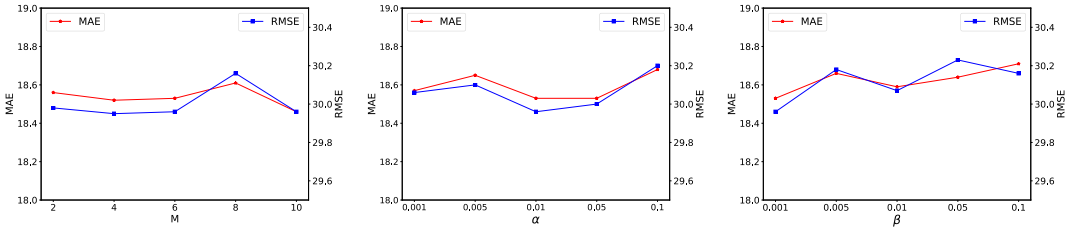
# Layers	$Encoder_{\mathbf{S}}$	$Encoder_{\mathbf{T}}$	$Decoder$	MAE	RMSE	MAPE (%)	$Steps_{\mathbf{S}}$	$Steps_{\mathbf{T}}$
				19.48	31.01	13.48	1.0	1.0
	✓			19.07	30.65	13.06	1.6	1.0
$L = 1$		✓		19.17	30.78	12.88	1.0	2.0
	✓	✓		18.97	30.44	**12.74**	1.6	2.1
	✓	✓	✓	**18.76**	**30.16**	12.91	1.2	3.7
				19.36	31.06	13.55	1.0	1.0
	✓			18.92	30.56	12.71	1.6	1.0
$L = 2$		✓		18.74	30.25	12.55	1.0	2.1
	✓	✓		18.70	30.15	**12.40**	1.2	4.0
	✓	✓	✓	**18.62**	**30.10**	12.78	2.5	2.8

of our model in coping with dynamic spatial-temporal task complexities. The DHM also benefits the decoder, which means the adaptive computation is also important for the accurate sequence reasoning. As to computation steps, our best model only employs 2.6 steps by average, while the performance is much better than the naive Transformer.



(a) MAE (b) RMSE (c) MAPE

Figure 3.4 : The performance comparison among different methods with varying horizons on PeMSD4 dataset w.r.t three metrics.



(a) Max recurrent steps (b) Loss weight for the recollection decoder (c) Penalty weight for the DHM

Figure 3.5 : The influence of three parameters w.r.t two metrics.

Parameter Analysis

To further investigate the robustness and effectiveness of our Bi-STAT method, we analyse the performance in terms of several related parameters.

Prediction performance w.r.t varying horizons. We compare with two competitive methods, GMAN and AGCRN, by varying the horizon from 1 to 12, and show the results in Figure 3.4. It can be observed that our method always outperforms the other two methods with a large margin. This demonstrates our method can generate accurate predictions for both the short-term streams (small horizon) and long-term streams (big horizon).

The robustness to hyper-parameters. We study the influence of three parameters, i.e max recurrent steps M, loss weight α for the recollection decoder, and penalty weight β for the DHM. M and β control the maximum computation load available to assgin to different tasks. α controls the regularization strength. From Figure 3.5, we can observe that our method is stable to the parameter changes. For each parameter, we choose the one that best reflects the performance-computation balance, e.g. $M = 6, \alpha = 0.01, \beta = 0.001$.

3.3.6 Visualization Results

In this subsection, we show some visualization results to provide a deep understanding of the mechanism of our method.

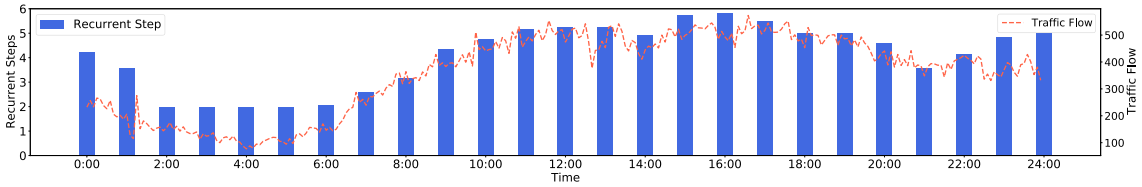


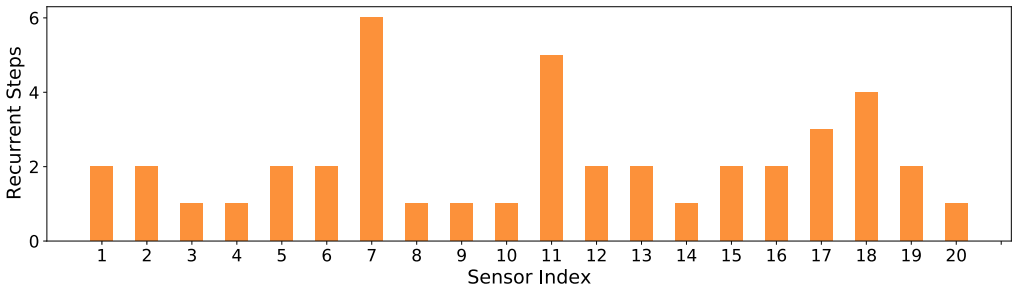
Figure 3.6 : The dynamic recurrent computation steps in different times (hours) of a day and the corresponding ground truth flow.

Dynamic computation w.r.t different spaces and times

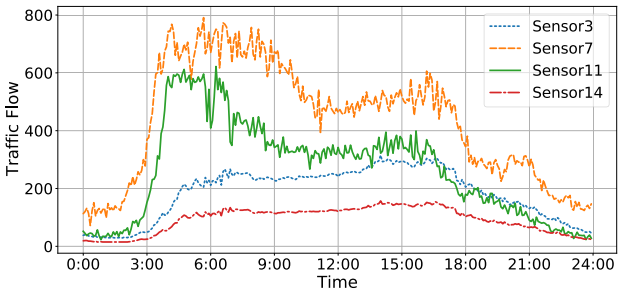
We first record all the recurrent computation steps of the spatial-adaptive Transformer and the temporal-adaptive Transformer during the testing process on the PeMSD8 dataset. Then, we post-process these recurrent steps for a better visualization. Temporally, we aggregate the recurrent steps of the temporal Transformer by hour for 24 hours in a specific day (i.e. Aug 20th, 2016). The recurrent steps are averaged for each hour and shown in the bar plot of Figure 3.6. Meantime, the corresponding ground truth flow is shown in the curve plot of Figure 3.6. It can be seen that the dynamic recurrent steps assigned to each hour maintains a high degree of consistency with the patterns of the ground truth flow (e.g. rush hour vs. off peak). The rush hours with more complex traffic patterns are assigned with more recurrent steps, while the off peaks with simple patterns are allocated with fewer recurrent steps. This indicates that our Dynamic Halting Module (DHM) indeed learns an effective model to adaptively assign the computation loads w.r.t the temporal task complexities.

Similarly, we also aggregate the recurrent steps of the spatial Transformer corre-

sponding to all sensors and the recurrent steps of some iconic sensors are shown in Figure 3.7 (a). It can be observed that most sensors have a relatively small recurrent steps (e.g. sensors 3 and 14). But there are also several sensors owing large recurrent steps (e.g. sensors 7 and 11). To further study the differences, we also plot the traffic flow of four representative sensors. Form the Figure 3.7 (b), we can observe that the oscillating patterns of sensor 7 and sensor 11 are much more complex than those of sensor 3 and sensor 14. In this sense, it is reasonable for sensor 7 and sensor 11 to obtain large computation loads. This visualization verifies the effectiveness of our model to assign dynamic spatial computation loads for different sensors.

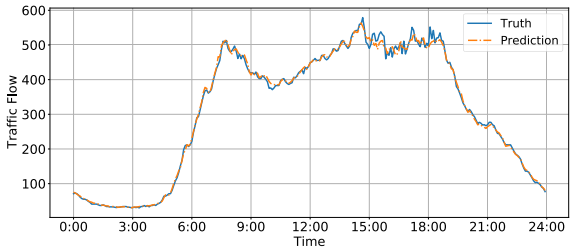


(a) Computation Steps

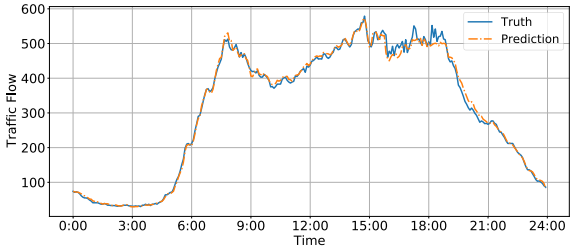


(b) Traffic Flow Truth

Figure 3.7 : Different number of computation steps for different sensors at the same time step and traffic flow truth for different sensors.



(a) 5min



(b) 60min

Figure 3.8 : The predicted traffic flow visualization on PeMSD4 for different prediction intervals (a.k.a horizon).

Visualization for both short-term and long-term prediction

We show the visualization results of the predicted traffic flow on PeMSD4 dataset for different prediction intervals in Figure 3.8. For both the short-term prediction (5min) and the long-term prediction (60 min), our method can accurately generate the prediction sequences. The visualization is consistent with our quantitative results in Table 3.2.

3.3.7 Computation Time

Finally, we compare the computation time of Bi-STAT with AGCRN, DCRNN and GMAN on the PeMSD4 dataset. For the comparison methods, we use their official code implementations. All methods are run on a server with Intel(R) Xeon(R) E-2288G CPU and a NVIDIA Corporation TU102GL GPU. It can be observed that AGCRN and DCRNN cost more computation time for both training and inference,

Table 3.6 : The computation time on the PeMSD4 dataset.

Model	Training (s/epoch)	Inference (s)
AGCRN	254.85	61.42
DCRNN	394.67	27.09
GMAN	230.55	11.41
Bi-STAT	239.21	13.33

due to the time-consuming process of the RNNs. In contrast, the transformer-based methods, i.e. GMAN and Bi-STAT (ours) show the substantially improved efficiency owning to the parallel and acceleration of the transformer architecture. Although Bi-STAT costs slightly longer time than GMAN, it endows the model with the adaptive and dynamic computation capability (by the well-devised DHM module). Therefore, Bi-STAT yields more accurate predictions than GMAN, e.g. 29.96 vs 31.06 (RMSE).

3.4 Conclusion

In this chapter, we propose a Bidirectional Spatial-Temporal Adaptive Transformer (Bi-STAT) for the spatial-temporal representation learning to deal with the urban traffic forecasting problem. Specifically, the model design is motivated by two intrinsic properties of the problem that are overlooked by existing methods. First, the imbalanced complexities of diverse forecasting problems require an adaptive computation model. Thus, we adopt the recurrent mechanism with a novel Dynamic Halting Module (DHM) to parsimoniously assign the essential computation load for each task. Second, the recollection of past traffic conditions plays an important role in the prediction of future traffic conditions. Therefore, we devise a recollection decoder to reconstruct the past traffic streams, which can provide extra context information for the future prediction task and serve as a regularizer to pre-

vent the future prediction task from fitting aggressively. All the designed modules are well incorporated to our Bi-STAT model. The experiments demonstrate the effectiveness of each model component and our Bi-STAT model outperforms all the state-of-the-arts with a large margin.

Chapter 4

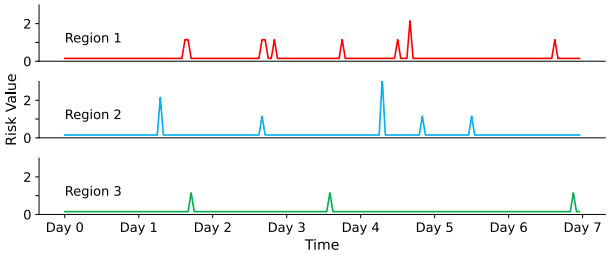
RiskContra: A Contrastive Approach to Forecast Traffic Risks with Multi-Kernel Networks

4.1 Introduction

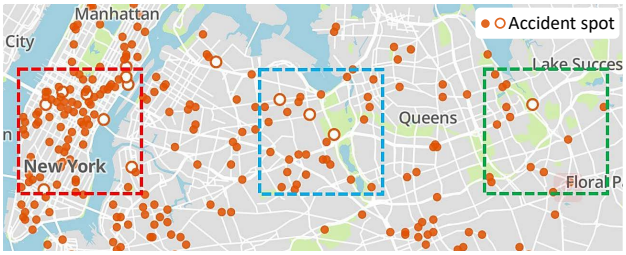
Traffic accidents have posed a serious threat to public safety due to the severe destructiveness and negative impact. The casualty from traffic accidents all over the world has amounted to over 1.3 million according to the World Health Organization (WHO) [72]. Thus, it is of vital importance to establish more effective traffic accident forecasting systems, which could significantly protect lives and properties from potential risks.

Traffic accident forecasting aims to predict the risk value of the latent traffic accident that may occur in a target region (*spatial*) at future time steps (*temporal*) given the historical traffic accident observations. Early works employ the statistical or linear machine learning models such as SVM or ARIMA [102] to explore the traffic accident patterns. However, they can hardly capture the complex and non-linear spatial-temporal correlations in the traffic accident data. Thus, recent works [68, 95, 97, 131] resort to deep learning models to promote the forecasting performance by taking advantage of the hierarchically deep and non-linear architectures, which have achieved significant progress. Despite the effectiveness of these deep forecasting methods, two challenges have greatly impeded the further improvement of existing traffic accident forecasting models.

The first challenge is the ultra-rareness of the accidents distributed across the temporal domain, i.e. the majority of the accidental risk value are zero due to the



(a) Ultra-rare temporal distribution



(b) Imbalanced spatial distribution

Figure 4.1 : Two intrinsic challenges of traffic accident forecasting.

infrequency of an accident (Fig. 4.1(a)). This hinders the effective training of deep neural networks, which requires sufficient examples to explore the data distribution and fit the model parameters. Moreover, since the number of non-risk samples is much larger than that of risk samples, the forecasting of traffic accidents will be easily biased to non-risks rather than potential risks. This temporal rareness issue, however, has not been well solved in previous works. To tackle this challenge, we devise a novel contrastive learning approach, which customizes the mixup [121] strategy to generate sufficient augmented risk samples for traffic accident forecasting. Our customized mixup strategy leverages the periodic patterns inherent in accident data. Specifically, we generate data augmentations by mixing two temporal samples with the same daily stamp from a fixed interval (e.g. one week). If either temporal sample is a risk sample, the generated sample will be labelled as risk with a mixing score. With these augmented risk samples, the ratio gap between the non-risk and risk samples is significantly reduced. Then, we utilize the generated

risk/non-risk samples to construct the positive and negative pairs, on which the supervised contrastive learning [50] is conducted. Equipped with contrastive learning, the feature embeddings of risk samples will be more distinguishable from those of non-risk samples, thus facilitating the accident forecasting performance.

The second challenge lies in the imbalanced spatial distribution of traffic accidents, which exhibits diverse patterns of multiple granularities from region to region, as shown in Fig. 4.1(b). For example, an urban area with crowded traffic flows may bear a higher risk compared to a rural counterpart with fewer vehicles, and a district with underdeveloped traffic management systems may experience more accidents than those equipped with advanced traffic monitoring systems. This challenge is not well addressed by existing works. GSNet [95] used the Convolution Neural Networks (CNNs) with a fixed kernel size to model the spatial correlation, which only captured the local and single-granularity features due to the limited receptive field. HeteroLSTM [118] modeled each region with a different ConvLSTM and adopt an ensemble strategy to generate the final results, which is infexible and parameter in-efficient. Different from these attempts, we design a delicate Multi-kernel CNNs architecture to capture the multi-granularity spatial correlations for traffic accident forecasting. In this architecture, the local, global, and point-wise convolution layers are elegantly arranged in a unified model (Fig. 4.2). The local convolution layer targets at regions with densely-distributed accidents, while the global convolution layer focuses on regions with sparsely-distributed accidents. Moreover, the point-wise convolution can conduct the dimension reduction to alleviate the computation burden caused by the feature aggregation, thus guaranteeing the efficiency of the multi-kernel networks.

4.2 Related Work

Contrastive Learning. Contrastive learning has shown remarkable successes in various domains [14, 15, 37]. It aims to distinguish the semantically similar samples (positive pairs) from the semantically dissimilar samples (negative pairs) in the latent space. Typically, SimCLR [14] employs data augmentations such as cropping, color distortion and blurring to generate different views of image as positives. MoCO [37] maintains the negative samples in a queue and employs the momentum encoder to ensure the consistency of the queue. SimSiam [15] explores a stop-gradient strategy to prevent contrastive learning from collapsing. Mixup [121] has proved to be an effective data augmentation strategy in supervised learning, which generates new samples by the convex interpolation of two different samples in both the data and label space. Albeit simple, mixup has been adopted as an effective regularization to mitigate the overfitting and improve the robustness of deep neural networks [48, 92, 93]. However, this effective strategy has not been studied in the context of traffic accident forecasting, which greatly suffers from the data lacking and overfitting issues. Motivated by this, we devise a customized mixup strategy to serve as an effective augmentation function in the proposed contrastive learning approach.

4.3 Methodology

4.3.1 Problem Definition

In the traffic accident forecasting task, we are given the historical traffic accident features $\{X_1, X_2, \cdots, X_T\}$, where $X_t \in \mathbb{R}^{W \times H \times D}$ represents D-dimensional traffic accident related features including accident risk, temporal features, POIs information, taxi order, and weather information at time step t in all regions (which are partitioned from grid map of the city under study with width W and height H).

In addition, temporal features E_{T+1} (hour in a day, day in a week, and holiday) of the future time step $T+1$ is given. The aim is to predict the future accidental risk values for all regions at time step $T+1$:

$$Y_{T+1} = \mathcal{F}_\theta(\{X_1, X_2, \cdots, X_T\}, E_{T+1}), \tag{4.1}$$

where θ is the model parameter, and $Y_{T+1} \in \mathbb{R}^{W \times H}$ denotes the predicted risk values for all regions.

4.3.2 Spatial-Temporal Accident Forecasting

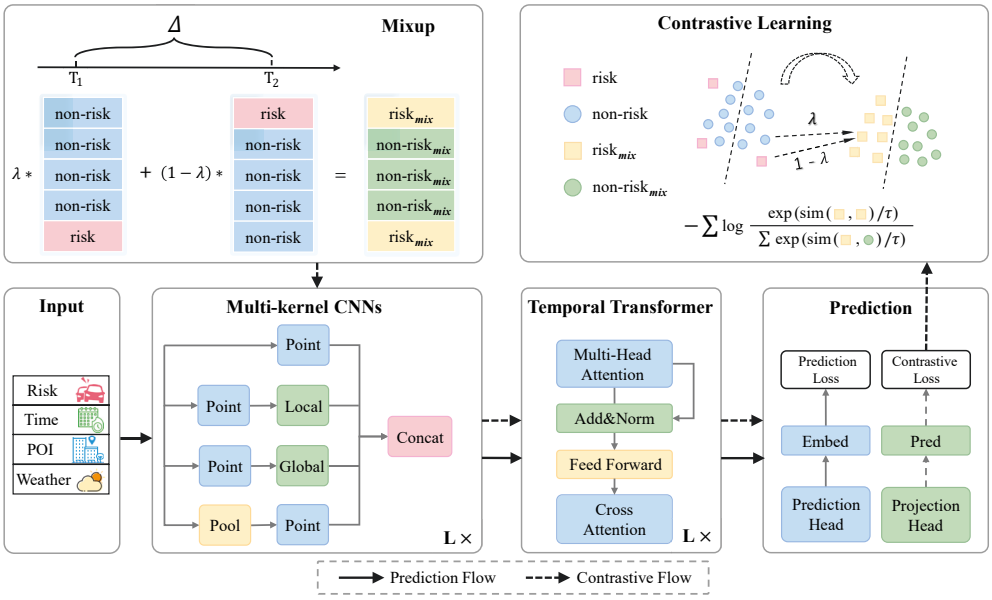


Figure 4.2 : The framework of RiskContra.

Multi-kernel Networks. Existing works employ CNNs, GCNs or Transformer to capture the spatial correlations in spatial-temporal tasks [34,40,95,98,130]. However, CNNs and GCN can only handle the local spatial proximity with a limited receptive filed, and Transformer is inefficient to model large number of elements. Taking both the model capacity and computation efficiency into account, we thus propose the Multi-kernel networks that can capture the hierarchical spatial correlations to address the spatial imbalance issue.

We define a single-layer convolution module as follows:

$$f(X; W, b) = \mathrm{ReLU}(\mathrm{W} * \mathrm{X} + \mathrm{b}), \tag{4.2}$$

where $*$ is the convolution operation, ReLU is the activation function, and W and b are the parameters. Our Multi-kernel networks are composed of diverse convolution modules targeting at regions with different spatial granularities (Fig. 4.2). We omit the layer index $l \in \{1, 2, \ldots, L\}$ for simplicity.

First, we adopt the ***Point-wise CNN*** $f_{\mathrm{P}}(\cdot)$ to capture the intra-region spatial representation. Besides, for the areas where risk is densely distributed across regions, the geographical proximity is a contributing factor of traffic accidents. For example, two spatially close roads in the busy commercial center tend to gain higher risks of traffic accidents. To capture local correlations among the adjacent regions such as road intersections and crowded streets, we design the ***Local CNN*** module $f_{\mathrm{L}}(\cdot)$ with *a small kernel size* to explore their *spatial proximity*.

Moreover, geographically faraway regions may share similar accident patterns due to similar POI distributions and temporal conditions. These semantic correlations are critical for analyzing accidents caused by external factors, which can significantly promote the forecasting performance. Thus, for these areas with sparsely distributed risks, a ***Global CNN*** module $f_{\mathrm{G}}(\cdot)$ with *a large kernel size* is devised in parallel to capture the spatial correlations among *distant regions*.

Except for the above parametric convolution modules, we also propose a ***non-parametric Pooling-based module*** $f_{\mathrm{pool}}(\cdot)$, which has the potential to avoid the overfitting issue of the parameterized modules.

To handle diverse regions, the concatenation operation Concat[$\cdot$] is adopted to fuse the output from different CNN modules to obtain a comprehensive feature representation. However, the concatenation operation would increase the feature dimension, thus inducing excessive computation burden especially for the CNN with

a large kernel size. To overcome this drawback, we employ another set of point-wise convolutions $f_{\mathrm{P}}(\cdot)$ to conduct the dimension reduction for computation efficiency.

At last, the output of the Multi-kernel Networks is represented as:

$$H = \mathrm{Concat}\big[\mathrm{f_P(X)};\, \mathrm{f_L(f_P(X))};\, \mathrm{f_G(f_P(X))};\, \mathrm{f_P(f_{pool}(X))}\big]\,. \tag{4.3}$$

Temporal Transformer and Cross-Attention. The traffic accident condition is tightly correlated to the historical observations. Both chronologically near (*short-term*) and periodically distant (*long-term*) time intervals have a significant impact on the target time step. Thus, we devise a Temporal Transformer module to dynamically capture both the long-term and short-term temporal correlations in traffic accident. Specifically, the multi-head self-attention [91] is adopted, followed by the dropout, layer normalization and a transition function (2-layer feed-forward network) to obtain a more powerful representation.

To predict the risk of a future time step based on the historical spatial-temporal features, we devise a ***cross-attention module*** to establish the correlation between future and historical sequences. Specifically, we employ temporal features E_{T+1} (time of a day, day of a week, and holiday) of the target time step $T+1$ as the query to explore its correlation with the spatial-temporal features H from Eq. 4.3 at each historical time step. The attention score between $T+1$ and each historical time step $t \in \{1, 2, \ldots, T\}$ is computed:

$$\mathrm{att} = \mathrm{Softmax}\left(\mathrm{ReLU}\left(W_H H + W_E E_{T+1} + b\right)\right)\,, \tag{4.4}$$

where $W_H \in \mathbb{R}^{1\times D_H}, W_E \in \mathbb{R}^{T\times D_T}$, and $b \in \mathbb{R}^T$ are weight and bias parameters. Then the future representation aggregates the most relevant historical features according to the attention score:

$$H_{T+1} = \sum_{t=1}^{T} \mathrm{att}_t \cdot H_t\,. \tag{4.5}$$

Prediction. For risk prediction, H_{T+1} is fed into a two-layer prediction head (PredHead) to generate the final forecasting results: $\hat{Y}_{T+1} = \text{PredHead}(H_{T+1})$.

4.3.3 Contrastive Learning with Mixup

Mixup as the Augmentation Function. To mitigate the ultra-rareness issue of the risk samples, we ***design a customized mixup strategy*** as the augmentation function to generate sufficient augmented risk samples. Inspired by the periodic temporal pattern that exists in traffic accident data, we generate the augmented samples through the *convex combination* of temporal samples $\mathbf{x}_i$ and $\mathbf{x}_j$ (with the same daily time stamp) from a fixed time interval Δ for both the features and labels as follows:

$$\mathbf{X}_{mix} = \{\lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j \,|\, t_i - t_j = \Delta\}, \tag{4.6}$$

$$\mathbf{Y}_{mix} = \{\lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j \,|\, t_i - t_j = \Delta\}, \tag{4.7}$$

where λ is the mixing coefficient to control the impact of two data sources. Δ reflects the periodic patterns, e.g. $\Delta = 1$ denotes daily pattern and $\Delta = 7$ denotes weekly pattern. If either $\mathbf{x}_i$ or $\mathbf{x}_j$ is a risk sample, then the generated sample will be labelled as a risk sample. The mixed data could: (1) inherit the specific pattern from the original accident data, with the mixed label simulating diverse risk values of the future occurrence of potential accidents; (2) maintain the temporal schemes such as hour of day and day of week in the original dataset, which is an important factor in the traffic forecasting task; (3) significantly bridge the ratio gap between the risk and non-risk samples to facilitate the risk representation in the following contrastive learning.

RiskContra. After setting up the effective mixup augmentation function, we devise a novel RiskContra approach that employs the augmented samples to tackle the temporal rareness challenge.

Firstly, we group the generated risk/non-risk samples into two disjoint positive and negative sets according to their mixed risk values. Specifically, the samples with the risk value larger than 0 are combined into the positive set, while those with the risk value equal to 0 are combined into the negative set.

Then, both the positive and negative sets are fed into an encoder $f(\cdot)$ composed of the Multi-kernel networks and the Temporal Transformer, and followed by a projection head $g(\cdot)$ to obtain the latent embedding. Formally, the embeddings of the positive and negative sets are produced as follows:

$$\mathbf{Z}_{pos} = \{g(f(\mathbf{x}_i)) \,|\, \mathbf{x}_i \in \mathbf{X}_{mix}, \mathbf{y}_i > 0\}\,, \tag{4.8}$$

$$\mathbf{Z}_{neg} = \{g(f(\mathbf{x}_i)) \,|\, \mathbf{x}_i \in \mathbf{X}_{mix}, \mathbf{y}_i = 0\}\,. \tag{4.9}$$

Finally, supervised contrastive learning is conducted in the latent embedding space to encourage the similarity of all positive pairs from risk embeddings while to discourage the similarity of all negative pairs:

$$\mathcal{L}_C = -\sum_{z_i \in \mathbf{Z}_{pos}} \log \frac{\sum_{\mathbf{z}_j \in \mathbf{Z}_{pos}} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{z_k \in \mathbf{Z}_{pos} \cup \mathbf{Z}_{neg}} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}\,, \tag{4.10}$$

where $\text{sim}(\cdot)$ is instantiated by the cosine similarity, and τ is the temperature.

Benefiting from the mixup strategy, more augmented risk samples could be utilized to prevent the contrastive learning from trivially overfitting the non-risk samples, which could significantly facilitate the accident forecasting performance by learning better risk representation.

4.3.4 Loss Function

At training time, the overall loss includes the accident prediction loss $\mathcal{L}_F$ and the contrastive loss $\mathcal{L}_C$. Specifically, $\mathcal{L}_F$ is computed as the mean square error between prediction results and the ground truth:

$$\mathcal{L}_F = \frac{1}{W}\frac{1}{H}\sum_{w=1}^{W}\sum_{h=1}^{H}\left(Y[w,h] - \hat{Y}[w,h]\right)^2\,, \tag{4.11}$$

where Y and $\hat{Y}$ respectively denote the true traffic accident risk and the predicted accident risk accident at target time $T+1$, and W and H are the width and height of the grid map covering all regions. The final loss is a combination of the two losses with a regularization parameter α to control the impact of contrastive learning: $\mathcal{L} = \mathcal{L}_F + \alpha\mathcal{L}_C$.

4.4 Experiments

Datasets and Evaluation Metrics. We employ two widely-used datasets for traffic accident forecasting, i.e. New York City (NYC) and Chicago [95]. We evaluate the proposed RiskContra by employing three commonly used metrics: Root Mean Square Error (***RMSE***), ***Recall*** and Mean Average Precision (***MAP***).

Baselines. To verify the effectiveness of the proposed RiskContra, we compare with 2 statistical methods: Historical Average (HA) and XGBoost [13], and 7 deep learning methods: MLP, GRU [20], SDCAE [10], ConvLSTM [84], Hetero-ConvLSTM [118], Graph WaveNet [106] and GSNet [95].

Implementation Details: For the Multi-kernel CNNs, the kernel-sizes (k) for point-wise, local, and global convolution module are set to 1×1, 3×3, and 5×5 respectively, which capture spatial correlations of multiple granularities to solve the spatial imbalance issue. The number of network layers L is set to 2.

In the mixup strategy, we set the interval $\Delta = 7$ to represent weekly patterns in accident occurance, weight $\lambda = 0.1$ to generate the augmented risk samples. As to the contrastive learning module, we set the temperature τ to be 0.4 for NYC and 0.8 for Chicago respectively, which can reflect the different accident conditions of the dataset. The weight α of the contrastive loss is set to $1e^{-5}$ and $1e^{-3}$ for NYC and Chicago respectively. The model and hyper-parameters are chosen according to the best performance on the validation set.

4.4.1 Comparison with the State-of-the-Art

Table 4.1 : Comparison with the state-of-the-art methods.

Model	NYC						Chicago					
	All Hours			Rush Hours			All Hours			Rush Hours		
	RMSE	Recall (%)	MAP	RMSE	Recall (%)	MAP	RMSE	Recall (%)	MAP	RMSE	Recall (%)	MAP
HA	10.3243	24.42	0.1049	9.4994	26.94	0.1258	14.9581	13.80	0.0572	10.2564	15.89	0.0644
XGBoost	11.0165	23.14	0.1008	10.173	25.22	0.1119	15.6946	12.58	0.0545	10.3685	15.22	0.0614
MLP	8.4289	27.28	0.1196	7.6379	29.51	0.1338	12.5116	17.53	0.0631	8.9500	18.93	0.0748
GRU	8.3375	28.09	0.1228	7.3546	30.76	0.1301	12.6482	17.83	0.0664	9.0421	18.66	0.0758
SDCAE	7.9774	30.81	0.1594	7.2806	31.22	0.1536	11.3382	18.78	0.0753	8.7543	20.58	0.1002
ConvLSTM	7.9505	30.99	0.1526	7.2554	32.61	0.1557	11.1309	18.84	0.0789	8.5254	20.30	0.0925
Hetero-ConvLSTM	7.9731	30.42	0.1454	7.275	31.43	0.1498	11.3033	18.43	0.0716	8.5437	18.93	0.0770
GraphWaveNet	7.7358	31.78	0.1623	7.0958	33.04	0.1647	11.0835	18.95	0.0805	8.4484	20.42	0.0933
GSNet*	7.6722	33.41	0.1856	6.8406	34.50	0.1797	10.8229	20.27	0.0976	8.2822	21.26	0.1204
RiskContra	**7.3994**	**34.48**	**0.1974**	**6.7103**	**35.54**	**0.1924**	**10.3784**	**22.30**	**0.1079**	**7.8663**	**23.87**	**0.1373**

The results are reported on two settings: (1) "**All Hours**" that measures the risk w.r.t all time intervals ranging from 0:00 to 24:00; (2) "**Rush Hours**" that considers the time intervals in 7:00-9:00 and 16:00-19:00, which are supposed to be the high-risk hours in a day.

The overall comparison is presented in Table 4.1. The proposed RiskContra outperforms all existing state-of-the-art models in terms of all metrics and datasets, demonstrating the better accident forecasting capability. Among all three metrics, RiskContra shows especially superior performance in terms of Recall and MAP. For example, the relative MAP improvement over the second best (i.e. GSNet) is 6.8% for "Rush Hours" on NYC. The relative Recall improvement for "All Hours" and "Rush Hours" on Chicago is 10.0% and 12.3%.

Notably, for all settings, the performance of "Rush Hours" is more superior than that of the "All Hours", which indicates that the higher ratio of risk samples in "Rush Hours" can substantially facilitate the model training. This is consistent

with our motivation to augment the risk samples for better contrastive learning.

4.4.2 Ablation Study and Visualization

Table 4.2 : Ablation study for Multi-kernel CNNs and Contrastive learning.

Model	All Hours			Rush Hours		
	RMSE	Recall (%)	MAP	RMSE	Recall (%)	MAP
CNN+TT	7.6044	33.16	0.1864	6.7569	33.62	0.1704
MKCNN+TT	7.5279	33.76	0.1912	6.7559	34.53	0.1861
CNN+TT+Contra	7.5424	34.02	0.1899	6.7201	34.08	0.1769
RiskContra	**7.3994**	**34.48**	**0.1974**	**6.7103**	**35.54**	**0.1924**

Ablation Study. To evaluate the effectiveness of our devised modules, we first construct a **baseline** by removing both the Multi-kernel CNNs and mixup contrastive modules from RiskContra, dubbed as "CNN+TT" (kernel size $k = 3$, TT stands for Temporal Transformer). Then, we add the Multi-kernel CNNs to the baseline and obtain the "MKCNN+TT" model. Similarly, "CNN+TT+Contra" stands for adding the mixup contrastive module to the baseline. RiskContra is our final model with both modules.

Ablation experiments are conducted on NYC dataset (Table 4.2). "MKCNN+TT" achieves significant improvements over the baseline "CNN+TT", corroborating the effectiveness of the Multi-kernel networks in capturing the multi-granular spatial correlations. "CNN+TT+Contra" also outperforms the baseline in all settings, which indicates that the augmented risk samples can facilitate the contrastive feature learning to improve accident forecasting performance. By combining all modules, the final model (RiskContra) achieves the best performance, showing the complementary effect of the two devised modules.

*Result obtained by running the official code with default parameter setting.

Table 4.3 : The impact of kernel size in the convolution layer.

Model	All Hours			Rush Hours		
	RMSE	Recall(%)	MAP	RMSE	Recall(%)	MAP
$CNN_{1\times1}$	7.8586	33.55	0.1866	7.0871	34.67	0.1856
$CNN_{3\times3}$	7.5424	34.02	0.1899	6.7201	34.08	0.1769
$CNN_{5\times5}$	7.5790	33.17	0.1775	6.7975	34.39	0.1736
$CNN_{7\times7}$	7.7378	31.38	0.1668	6.8653	31.95	0.1545
RiskContra	**7.3994**	**34.48**	**0.1974**	**6.7103**	**35.54**	**0.1924**

Table 4.4 : Comparisons of various mixup constractive variants.

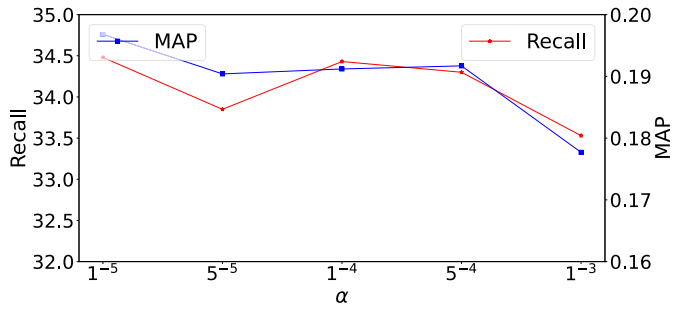
Model	Mixup	Contra	Supervise	All Hours			Rush Hours		
				RMSE	Recall(%)	MAP	RMSE	Recall(%)	MAP
A	✗	✓	✓	7.5837	34.24	0.1885	6.8012	34.64	0.1777
B	✓	✗	✓	7.4852	32.87	0.1854	6.7366	33.90	0.1754
C	✓	✓	✗	7.6113	34.00	0.1879	6.7679	34.78	0.1836
$RiskContra_{\Delta1}$	✓	✓	✓	7.4082	33.87	0.1909	**6.6443**	34.39	0.1809
$RiskContra_{\Delta7}$	✓	✓	✓	**7.3994**	**34.48**	**0.1974**	6.7103	**35.54**	**0.1924**

Model Variants. To further investigate each component in our RiskContra respectively, we implement various variants to analyse the specific module design.

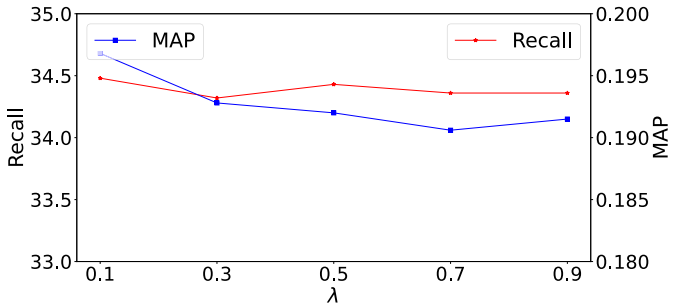
To study the impact of kernel size in convolution layers, we construct several model variants each with a different fixed kernel size (from $\{1\times1, 3\times3, 5\times5, 7\times7\}$) , namely $CNN_{1\times1}, \ldots, CNN_{7\times7}$. From Table 4.3, we can conclude that no single kernel size can obtain better performance than Multi-kernel CNNs. This verifies the necessity of Multi-kernel CNNs in tackling the spatial imbalance.

To show the rationale of our mixup contrastive framework, we construct several mixup and contrastive variants in Table 4.4. Model **A** applies contrastive learning on original data from two consecutive weeks without the mixup augmentation. The inferior performance compared with RiskContra implies that the contrastive learning alone can hardly improve the accident forecasting, which is due to the lack of

risk samples in contrastive pair construction. Model **B** utilizes the mixup data to compute a prediction loss instead of contrastive loss. The worse results show the advantage of contrastive mechanism over the redundant prediction task. Model **C** applies contrastive learning on the mixed data and the original data in an unsupervised manner without using the mixed label. The degraded performance verifies the importance of the virtual labels created by the mixup strategy to provide prior cues about risk/non-risk samples. Finally, ***RiskContra*** utilizes both the Mixup and Supervised Contrastive modules. To further verify the influence of the periodicity when performing the mixup strategy, we report the settings of both $\Delta = 1$ and $\Delta = 7$, representing daily and weekly periodicity, respectively. The results show that weekly mixup ($\Delta = 7$) achieves better performance compared to the daily counterpart in risk forecasting, which is consistent with the risk occurrence pattern in real life.



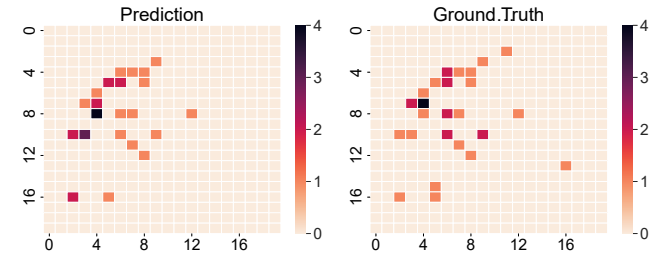
(a) Loss weight α



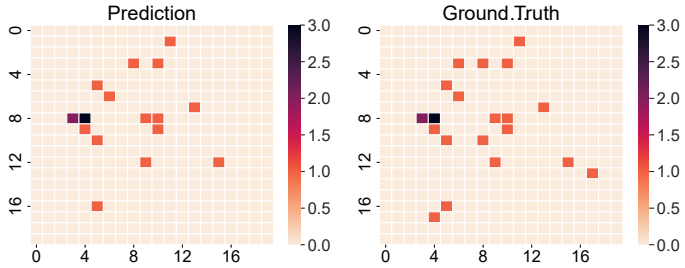
(b) Mix weight λ

Figure 4.3 : The influence of two important parameters w.r.t two metrics.

Parameter Analysis. We examine the impact of two important hyper-parameters, i.e. the loss weight α and the mixup weight λ. It can be observed from Fig. 4.3 that our model exhibits the robust performances with parameters varying in a reasonable range.



(a) Time: 11:00



(b) Time: 15:00

Figure 4.4 : The visualization of prediction results compared to the ground truth.

Visualization. To understand the effectiveness of RiskContra from a more intuitive perspective, we conduct a visualization of one-hour prediction results on the NYC dataset at different time. The heatmaps in Fig. 4.4 demonstrate the risk values at different regions with different colors, which are masked by the high risk region map to reflect more significant prediction results. It can be observed that the visualization results are generally consistent between the predicted traffic accident risks and the ground truth, which indicates the capability of our model to accurately forecast the future traffic risks.

4.5 Conclusion

This chapter proposes to tackle two intrinsic challenges in spatial-temporal representation learning for the traffic accident forecasting task. To deal with the spatial imbalance, we design the Multi-kernel Networks to capture the hierarchical spatial correlations among disparate regions. To address the temporal rareness, we devise a customized mixup strategy to generate sufficient augmented risk samples for effective contrastive learning. All proposed modules are incorporated into a novel contrastive learning approach with extensive experiments corroborating its effectiveness.

Chapter 5

Multivariate Traffic Demand Prediction via 2D Spectral Learning and Global Spatial Optimization

5.1 Introduction

Traffic demand prediction [2, 110, 112–114] is a critical component in the development of Intelligent Transportation System (ITS). The accurate prediction could help the traffic management system pre-allocate transportation resources to meet various traffic demands, which can significantly contribute to better commuting and travel services.

Traffic demand forecasting aims to forecast demand variables across multiple traffic stations in the city using historical records. These variables include pick-up and drop-off values for transportation requests such as taxi orders and shared bikes. It is a complex multivariate forecasting task that involves analyzing spatial-temporal patterns for each variable and the correlation among variables.

Recently, deep learning has become the mainstream approach for spatial-temporal traffic forecasting, due to its capability of modeling complex and non-linear spatial-temporal correlations in traffic data. Specifically, CNNs can model the regular spatial relations in partitioned traffic grids [95]. RNNs are capable of learning evolving patterns [114]. GCNs regard the traffic network as a graph to model the non-Euclidean correlations [3]. Transformer uses the self-attention mechanism to selectively extract the most informative features in the traffic context [127]. Despite these successes, recent studies [7, 78] find the spectral bias of deep networks: *deep*

networks tend to learn low-frequency functions that vary globally with fewer local variations. In the traffic forecasting context, low-frequency components represent the smooth trend of traffic patterns, while high-frequency ones reflect the transient events such as acute demand caused by accidents. Thus, traffic demand prediction will be hindered without sufficiently exploring the high-frequency traffic patterns.

In this chapter, we propose a novel embedded 2D Fourier learning framework to overcome the spectral bias issue in the deep learning-based traffic forecasting task. In our model, both low-frequency and high-frequency signals are effectively embedded and processed to capture comprehensive traffic patterns for accurate prediction. *Firstly*, we devise a ***spectral embedding function*** to transform the input variables to a higher dimensional space. Specifically, we adopt a series of sine and cosine functions to capture diverse frequency components of the multivariate traffic data. The amplitude of signals is adjusted by multiplying a matrix sampled from the Gaussian distribution. The spectral embedding function induces a well-behaved stationery (shift-invariant) kernel space, which preserves the distance properties of the original data. It also explicitly exposes both the low- and high-frequency variates of input to the model by regulating the band of the frequency in the sine and cosine functions. This can enable the deep neural networks to learn not only the basic trend but also the detailed variations of traffic patterns. Moreover, the embedded variables are endowed with rich spectral information to be readily leveraged in the subsequent Fourier processing.

Secondly, we model the ***temporal and multivariate interactions*** (e.g., the correlation among different traffic demand features) independently and develop the ***2D spectral representation learning*** along the two different dimensions. Several existing time-series methods directly apply 1D Fourier transformation on the temporal sequence [103, 124, 125, 129]. This is straightforward because traffic sequences naturally contain periodic patterns for Fourier analysis in the temporal

dimension. However, due to the implicit and in-direct multi-variable correlations, no existing method applies 2D Fourier analysis for simultaneous temporal and multivariate correlation modeling. In our framework, the spectral embedding function maps the multivariate features into diverse frequency components, which makes 2D spectral analysis a well-suited model. Therefore, the Fourier operations can be applied to the two dimensions, followed by the frequency domain representation learning to capture temporal patterns as well as multivariate correlations for accurate prediction.

Another important aspect of multivariate traffic demand prediction is spatial correlation modeling. The demand for specific transportation is distributed across the targeted city, with the data at each traffic station closely intertwined with each other. Therefore, it is a citywide task that should take into account the global structure and mutual constraints of the traffic network. Existing methods such as CNNs [53] and GCNs [51] model the spatial correlations through the local neighboring aggregation. However, the limited receptive field failed to capture the correlation from the globally similar tokens in the traffic network. Although the Transformer based methods [91] could dynamically attend to every token regardless of the relative distance, they can only conduct the pair-wise similarity computation individually without considering the mutual constraints from other pairs of correlation. All of these methods can not leverage the intrinsic global structure of the whole traffic network, which leads to inferior forecasting performance.

In this chapter, we ***formulate spatial correlation learning as an optimal transport problem***. Instead of directly utilizing the pairwise similarities, we optimize the objective function by maximizing the total spatial correlations in the citywide traffic network. To further avoid the possible trivial solutions, we impose a row and column sum constraint on the objective to induce a more balanced correlation map. In this way, the performance of prediction could be promoted by

exploiting the globally optimal spatial correlations.

5.2 Related Work

Spectral Representation Learning Fourier transform is a commonly effective technique for digital signal processing. Recently, discrete Fourier Transform (DFT) has been actively applied to deep neural networks to conduct spectral representation learning, with successful applications in various fields such as computer vision and natural language processing. Motivated by the spectral transform theory, [18] designed a Fourier Unit to perform convolution in the frequency domain, which leveraged both spatial and spectral information to enlarge the receptive field. [80] proposed global filters to learn the frequency domain features by depthwise global convolution. [32] learned to mix the feature tokens in the frequency domain. Apart from convolution, the parameter-free DFT operation has become a competitive alternative to self-attention in capturing correlations among different tokens. For example, [54] proposes to replace the self-attention mechanism with the Fourier Transform, which achieved comparable performance with much faster computation speed. Besides, there is also work intending to refine the underlying structure and optimization of frequency domain models. [74] proposes to perform only a single Fourier transform and optimize the approximation function directly in the frequency domain, which significantly streamlines the redundant forward and inverse Fourier transform in each layer of the model.

In the domain of time series forecasting, the learnable Fourier layers have also been incorporated to explore periodic patterns from the frequency perspective. Specifically, FEDformer [129] formulated time series as a sparse representation of Fourier transforms and developed the frequency domain self-attention mechanism. [103] designed a contrastive loss in the frequency domain to encourage discriminative seasonal representation. [125] proposed to enforce the time-frequency

consistency by encouraging the similarity between the time and frequency domain representation. [19] leveraged random Fourier features as an approximation of softmax kernel to linearize self-attention. [104] proposed an auto-correlation mechanism motivated by the time series periodicity and calculated the time-delay similarities by using a fast Fourier transform. [124] incorporated the frequency components into each hidden state of the recurrent neural networks to learn multi-frequency trading patterns in the stock sequences.

Although having achieved remarkable successes, the research on Fourier domain time series forecasting is limited to the temporal dimension, leaving the interactions among multivariate features unexplored. In this chapter, we propose a 2D Fourier model to explore the spectral interactions from both multiple time steps and variates. We also incorporate a spectral embedding function to promote the ability of deep neural networks to learn the high-frequency components of traffic inputs.

5.3 Methodology

5.3.1 Problem Definition

This chapter targets the multi-horizon and multivariate traffic demand forecasting task based on the overall traffic network, where the horizon represents the time step in traffic sequences. Specifically, we are given the historical traffic demand sequence $X \in \mathbb{R}^{P \times N \times d}$, which is collected from N stations of the past P horizons and contains d-dimensional multivariate features. Besides, we also have the spatial encoding SE and temporal encoding TE to serve as the external input. Specifically, SE is constructed from the similarity graph by the node2vec [127]. TE concatenates the time-of-day vector and day-of-week vector which contains the specific normalized values indicating the 24 hours in a day and the 7 days in a week.

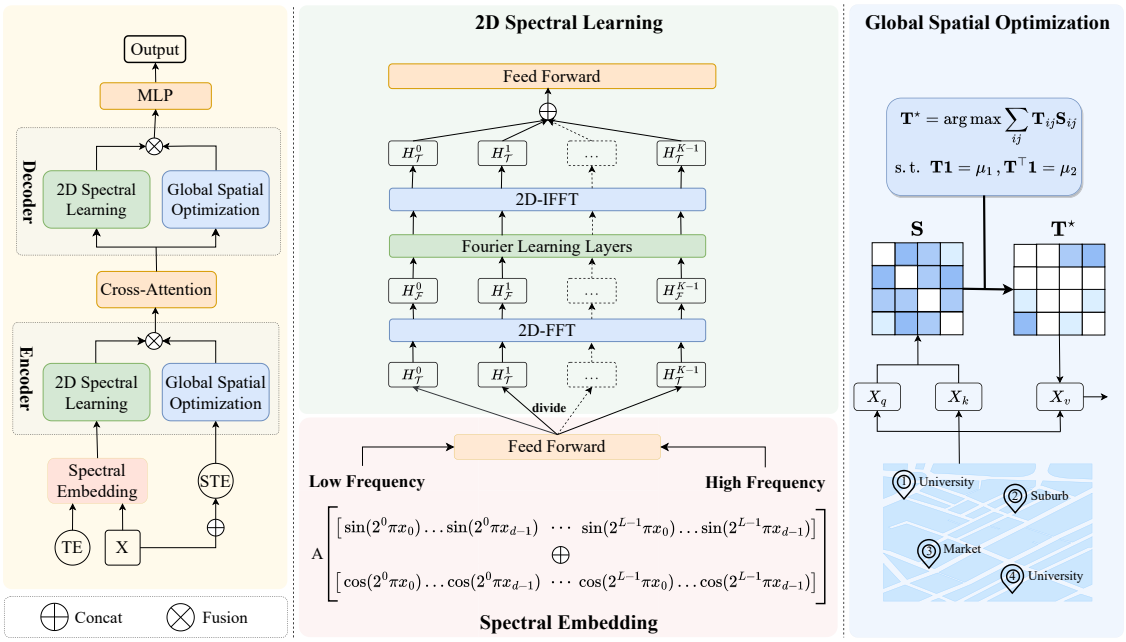


Figure 5.1 : The framework of our method. It follows an encoder-decoder architecture, where both the encoder and decoder consist of a 2D spectral representation learning and global spatial optimization module, with spectral embedding applied to enable the learning of both low-frequency and high-frequency components of the input.

The aim is to predict the traffic demand of the future Q horizons $Y \in \mathbb{R}^{Q\times N\times d}$:

$$Y = \mathscr{F}_\theta(X; SE, TE)\,, \tag{5.1}$$

where $\mathscr{F}$ is the learnable model with parameter θ.

5.3.2 Embedded 2D Spectral Learning

Discrete Fourier Transform

Discrete Fourier Transform (DFT) is the building block of our model for Fourier learning. Specifically, given a sequence $\{x_n\}_{n=0}^{N-1}$ in the time domain, DFT converts it to the frequency domain as follows:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j(\frac{2\pi}{N})kn}\,, \tag{5.2}$$

where j is the imaginary symbol, and X_k represents the k-th spectral component in the frequency domain with the frequency $\omega_k = 2\pi k/N$. Similarly, inverse DFT converts a sequence in the frequency domain back to the time domain:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j(\frac{2\pi}{N})kn} . \tag{5.3}$$

In practice, Fast Fourier Transform (FFT) algorithm is usually employed to perform DFT efficiently.

Spectral Embedding Function

Spatial-temporal methods typically adopt MLP layers to project the input variable to a higher dimension, before feeding it to the prediction model. However, recent studies found the spectral bias issue [7,78], i.e., neural networks tend to learn a low-frequency function and ignore the high-frequency components. Therefore, the detailed variations of the input sequence cannot be easily captured by these models for accurate prediction.

To overcome this issue, we design an effective spectral embedding function to encapsulate both the low-frequency and high-frequency components of the original signal. The embedding function uses the sine and cosine functions to extract multiple frequency components of the multivariate input data. Given $\mathbf{x} \in \mathbb{R}^d$, a function γ maps it to the $\mathbb{R}^{2dL}$ feature space as follows:

$$\gamma_{\sin}(\mathbf{x}) = [\sin(2^0\pi\mathbf{x}), \ldots, \sin(2^{L-1}\pi\mathbf{x})] \in \mathbb{R}^{dL} , \tag{5.4}$$

$$\gamma_{\cos}(\mathbf{x}) = [\cos(2^0\pi\mathbf{x}), \ldots, \cos(2^{L-1}\pi\mathbf{x})] \in \mathbb{R}^{dL} , \tag{5.5}$$

$$\gamma(\mathbf{x}) = \mathbf{A}[\gamma_{\sin}(\mathbf{x}), \gamma_{\cos}(\mathbf{x})] \in \mathbb{R}^{2dL} , \tag{5.6}$$

$$\mathbf{A} \sim \mathcal{N}(\mu, \sigma^2) . \tag{5.7}$$

Here, L is the embedding length to regulate the range of frequency. The introduction of the matrix $\mathbf{A}$ is motivated by random Fourier features [79]. $\mathbf{A}$ can be used as a scaling coefficient to adjust the amplitude of embedded Fourier features.

Proposition 1. Using the embedding function $\gamma(\mathbf{x})$, a stationary (shift-invariant) kernel $\mathbf{K}$ can be constructed as follows:

$$\begin{aligned}\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2) &= \gamma(\mathbf{x}_1)^\top \gamma(\mathbf{x}_2) \\ &= \mathrm{Tr}(\mathbf{A}^\top \mathbf{A}) \left(\sum_{i=1}^{d} \sum_{\ell=1}^{L} \cos\left(2^\ell \pi (\mathbf{x}_1^{(i)} - \mathbf{x}_2^{(i)})\right) \right).\end{aligned}$$

Proposition 1 demonstrates that our embedding function preserves the good distance property of the original data, which can be fully utilized to explore the spatial-temporal correlation for accurate prediction. More importantly, it extracts all frequencies (low, medium and high) of the input data, alleviating the spectral bias issue of deep networks.

Besides traffic demand sequences, the temporal context information (e.g., time of day and day of week) is an important prior for the prediction task. For example, rush hours tend to have higher demands. Previous works usually embed this information with a one-hot vector, followed by an MLP projection. However, as mentioned before, this strategy suffers from spectral bias and is inconsistent with our Fourier features. So, we embed the temporal encoding (TE) in a similar way to $\mathbf{x}$ with a different embedding length M:

$$TE_{\text{embed}} = [\gamma_{\sin}(TE), \gamma_{\cos}(TE)]\,, \tag{5.8}$$

$\mathbf{A}$ is omitted here because the temporal context has a fixed range of values indicating the relative time in a day and a week.

Finally, the features from the traffic sequence and the temporal encoding are combined to produce the final spectral features:

$$H_{\mathcal{T}} = f(\gamma(\mathbf{x}) || TE_{\text{embed}}) \in \mathbb{R}^{T \times N \times D}\,, \tag{5.9}$$

where T and D are the temporal and the hidden dimension respectively, $||$ denotes the concatenation operation, and $f(\cdot)$ is a multi-layer feed-forward network to fuse the two sources of feature embeddings.

2D Spectral Learning

After obtaining the embedded features $H_{\mathcal{T}}$, a straightforward approach is to directly apply 1D-FFT on the time dimension for spectral representation learning [129]. However, the prediction of each variable relies on not only historical records temporally but also other variables in the multivariate prediction task. The cross-dimension correlations of multiple feature variables also evolve over time, which makes it possible for the spectral learning to extract these dynamic patterns. Thus, we regard the multiple variables as a sequential signal to explore the spectral interactions among different variates.

With the explicit temporal correlations and implicit variable interactions, we devise a 2D-FFT based method to perform frequency domain representation learning. Specifically, we borrow the idea from the multi-head attention to conduct multi-spectrum learning in the frequency domain. Firstly, we divide the spectral embedding $H_{\mathcal{T}} \in \mathbb{R}^{T\times N\times D}$ into K blocks along the feature dimension, with each block $H_{\mathcal{T}}^{k} \in \mathbb{R}^{T\times N\times \frac{D}{K}}$ representing the diverse subspaces of the spectral embedding. Then, we apply the 2D-FFT to each block to independently map the features from the time domain to the frequency domain:

$$H_{\mathcal{F}}^{k} = \mathcal{F}_{T}(\mathcal{F}_{D}(H_{\mathcal{T}}^{k}))\,, \tag{5.10}$$

where $\mathcal{F}$ represents the FFT transformation with the subscript denoting the specific transformation dimension (i.e., feature dimension D and time dimension T).

Then, in each embedding block, we construct a Fourier learning layer to extract the distinctive features in the frequency domain:

$$H_{\mathcal{F}}^{k} = \sigma(\mathrm{abs}(W_{\mathcal{F}}^{k} H_{\mathcal{F}}^{k}))\,, \tag{5.11}$$

where $\sigma(\cdot)$ denotes the non-linear activation function, and abs represents the real component of the complex frequency features, which is motivated by [54] to avoid

the modifications of the feed-forward layers. The Fourier learning layers modulate the amplitude and phase components of spectral features through training multiple complex weights $W_{\mathcal{F}}^{k}$ to obtain a desired representation in the inverse time domain for better forecasting performance.

After the Fourier learning, an inverse 2D-FFT is applied to transform the modulated frequency domain features back to the time domain:

$$H_{\mathcal{T}}^{k} = \mathcal{F}_{T}^{-1}(\mathcal{F}_{D}^{-1}(H_{\mathcal{F}}^{k}))\,, \tag{5.12}$$

where $\mathcal{F}^{-1}$ is the inverse FFT.

Finally, the inversely transformed feature blocks are concatenated together and fed into a time domain feed-forward network $g(\cdot)$ to complete the alternating time and frequency learning:

$$H_{\mathcal{T}} = g(\text{Concat}(H_{\mathcal{T}}^{0}, \ldots, H_{\mathcal{T}}^{K-1}))\,. \tag{5.13}$$

Compared to conventional time-domain models such as CNN, RNN, and Transformer which have difficulties learning the high-frequency patterns [78], the proposed spectral learning method benefit from directly operating on the frequency domain to capture both low-frequency and high-frequency patterns for accurate prediction.

5.3.3 Global Spatial Optimization

In the traffic demand prediction task, the future passenger demand in a specific region not only depends on the historical records but also correlates with other regions sharing similar properties, such as the distribution of Points of Interest (POI) and the semantic function of the region. Existing works model this spatial correlation with CNN, GCN, or Transformer. We argue that these methods can only calculate the local or pairwise correlations and are uninformed of the potential global structure of the traffic network. Thus, they can not leverage the mutual constraint in the traffic network to help achieve the globally optimal forecasting performance.

In this chapter, we consider the citywide traffic network and formulate spatial correlation learning as a global optimal transport problem. Firstly, we follow the pairwise matching mechanism in self-attention to extract all possible spatial relationships in the traffic network by obtaining a similarity matrix $\mathbf{S}$. Specifically, we concatenate demand features X, the spatial-temporal encoding STE (which is constructed from SE and TE) to serve as input. Then, three different MLP layers are applied respectively to obtain the query X_q, key X_k, and value X_v. The matrix $\mathbf{S}$ is calculated as:

$$\mathbf{S} = \frac{X_q \cdot X_k^T}{\|X_q\|\|X_k\|} \,. \tag{5.14}$$

In previous methods, $\mathbf{S}$ is directly used as the spatial correlation to conduct the representation learning, such as in Spatial Transformer [127]. However, this strategy is suboptimal for two reasons: (1) the individual elements in $\mathbf{S}$ do not take into account the overall structure and the mutual constraints in the traffic system, (2) the noisy and weak correlations will harm the future prediction by occupying the intrinsically important correlations.

To overcome these disadvantages, we propose an optimal transport objective function based on $\mathbf{S}$ to obtain an optimized global spatial correlation instead of directly applying it. First, we define a matrix $\mathbf{T}$ to represent the probability of the correlation between the query and key. Then, we formulate the following optimization function aiming to maximize the total correlations from all pairs of connections in the whole traffic network. Specifically, the optimization problem is defined as:

$$\begin{aligned} \mathbf{T}^\star &= \arg\max \sum_{ij} \mathbf{T}_{ij}\mathbf{S}_{ij} \,, \\ \text{s.t. } & \mathbf{T}\mathbf{1} = \mu_1 \,, \mathbf{T}^\top \mathbf{1} = \mu_2 \,. \end{aligned} \tag{5.15}$$

Here, the row and column sum constraints are imposed on T, and μ_1 and μ_2 are two distribution probability vectors to avoid trivial solutions such as all values being 0 except for the maximum. We set the values of μ_1 and μ_2 as $\frac{1}{N}$, assuming the

balanced correlation assignment.

We find that our global optimization function can be easily converted to a standard optimal transport problem [94] introduced above by defining the cost matrix $C = 1 - S$ as the dissimilarity of the traffic network. Solving problem (5.15) is time-consuming, we follow the general practice to solve an entropy regularized problem using the Sinkhorn-Knopp algorithm [22] as follows:

$$\begin{aligned} \mathbf{T}^{\star} = \arg\min \sum_{ij} \mathbf{T}_{ij}\mathbf{C}_{ij} + \alpha H(\mathbf{T}) \,, \\ \text{s.t.} \;\; \mathbf{T}\mathbf{1} = \mu_1 \,, \mathbf{T}^{\top}\mathbf{1} = \mu_2 \,, \end{aligned} \tag{5.16}$$

where $\mathbf{C} = 1 - \mathbf{S}$, $H(\mathbf{T}) = \sum_{ij} \mathbf{T}_{ij}(\log \mathbf{T}_{ij} - 1)$ is the negative entropy, and α is the regularization parameter.

Finally, we employ $\mathbf{T}^{\star}$ as the attention matrix to selectively obtain the spatial output according to the importance of globally optimal correlations:

$$H_S = \mathbf{T}^{\star} X_v \,. \tag{5.17}$$

By considering the global structure and the mutual constraints of the overall traffic network to obtain an optimal correlation matrix $\mathbf{T}^{\star}$, our model can take full advantage of the spatially similar regions to promote the performance of traffic demand prediction.

5.3.4 Overall Architecture and Loss Function

The overall framework is shown in Figure 5.1, which integrates both the frequency representation learning and global spatial optimization modules in an encoder-decoder architecture. A cross-attention module is designed to bridge the encoder and decoder, which maps the historical traffic features to predict future demands. Each layer of the encoder and decoder shares the same structure of the spatial-temporal modules, the outputs of which are fused at the end of the layer to

capture the entangled spatial-temporal correlations. Multiple encoder and decoder layers are stacked to learn more powerful feature representations.

The loss function is defined as the Root Mean Square Error (RMSE) between the predicted and true traffic demand values among the N traffic stations and Q future horizons:

$$\mathcal{L} = \sqrt{\frac{1}{N}\frac{1}{Q}\sum_{n=1}^{N}\sum_{t=1}^{Q}(Y_{[n,t]} - \hat{Y}_{[n,t]})^2}\,. \tag{5.18}$$

5.4 Experiment

5.4.1 Datasets

To demonstrate the effectiveness of our method, we conduct experiments on two real-world traffic datasets collected by NYC OpenData*. The two datasets record the pick-up and drop-off traffic order information of taxis and bikes in NYC respectively. Detailed information is as follows:

- NYC-Bike: it collects demand for sharing bikes at 250 stations according to the daily use in NYC. The time span is chosen from April 1st, 2016 to June 30th, 2016.
- NYC-Taxi: it contains taxi demand information collected from 266 traffic stations with time spanning from April 1st, 2016 to June 30th, 2016.

5.4.2 Baseline Methods

We compare our method with the following baselines, including both statistical and deep learning based models:

- HA: Historical Average, which takes the average of the historical traffic demand values as the forecasting result.

*https://opendata.cityofnewyork.us/

- XGBoost [13]: which is an efficient tree boosting based system with broad applications in machine learning.
- FC-LSTM [38]: which is the Long Short Term Memory combined with the fully connected network for sequence prediction.
- DCRNN [60]: which incorporates the diffusion convolution into the Gated Recurrent Unit (GRU) in an auto-encoder architecture.
- STGCN [115]: which employs the graph convolutional and causal convolutional layers to model the spatial temporal interdependencies.
- STG2Seq [2]: which designs a hierarchical GCN structure to learn the spatial correlations and an attention mechanism to learn from historical features.
- GraphWaveNet [106]: which is constructed by the GCN and the gated temporal convolution network (Gated TCN) to model the spatial-temporal graph information.
- MTGNN [105]: which learns an adjacency matrix to capture the spatial relationships among time series.
- STSGCN [86]: which synchronously capture the localized spatial-temporal correlations using the graph convolutional module.
- AGCRN [3]: which can capture node-specific spatial and temporal correlations in the traffic series.
- GMAN [127]: which constructs multiple layers of spatial-temporal attention blocks in an encoder-decoder architecture to extract the spatial-temporal correlations.

- STTN [107]: which proposes Spatial-Temporal Transformer Networks to jointly capture the spatial-temporal dependencies for long-term traffic forecasting.
- Traffic Transformer [9], which employs GCN and transformer respectively to capture the spatial-temporal dependencies in traffic data.
- CCRNN [114]: which incorporates different self-learned adjacency matrices into each layer of the GCN model.
- ESG [112]: which learns the evolving and multi-scale interdependencies in time series by constructing different adjacency matrices in the GRU model.

5.4.3 Evaluation Metrics

To evaluate the performance of the baselines and our method, three commonly-used metrics are adopted, i.e., Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Pearson Correlation Coefficient (PCC) [114]. Specifically, they are defined in the following:

$$\text{RMSE} = \sqrt{\frac{1}{N}\frac{1}{T}\sum_{n=1}^{N}\sum_{t=1}^{T}(Y_{[n,t]} - \hat{Y}_{[n,t]})^2}\,, \tag{5.19}$$

$$\text{MAE} = \frac{1}{N}\frac{1}{T}\sum_{n=1}^{N}\sum_{t=1}^{T}\left|Y_{[n,t]} - \hat{Y}_{[n,t]}\right|, \tag{5.20}$$

$$\text{PCC} = \frac{1}{N}\sum_{n=1}^{N}\frac{\sum_{t=1}^{T}(Y_{[n,t]} - \bar{Y}_{[n]})(\hat{Y}_{[n,t]} - \bar{\hat{Y}}_{[n]})}{\sqrt{\sum_{t=1}^{T}(Y_{[n,t]} - \bar{Y}_{[n]})^2}\sqrt{\sum_{t=1}^{T}(\hat{Y}_{[n,t]} - \bar{\hat{Y}}_{[n]})^2}}\,, \tag{5.21}$$

where N is the number of traffic stations, T is the number of time steps (horizons), Y is the ground truth value ($\bar{Y}$ is the mean), $\hat{Y}$ is the predicted value ($\bar{\hat{Y}}$ is the mean). The first two metrics measure the error between the prediction and the ground truth, with a lower value representing better performance. The last metric is used to measure the linear correlation between the predicted and ground truth sequences, with a larger value representing better performance.

Table 5.1 : Traffic demand forecasting results on two datasets. The best performance in each column is highlighted in boldface.

Method	NYC-Taxi			NYC-Bike		
	RMSE	MAE	PCC	RMSE	MAE	PCC
HA	29.7806	16.1509	0.6339	5.2003	3.4671	0.1669
XGBoost [13]	21.1994	11.6806	0.8007	4.0494	2.4690	0.4861
FC-LSTM [38]	18.0708	10.2200	0.8645	3.8139	2.3026	0.5675
DCRNN [60]	14.7926	8.4274	0.9122	3.2094	1.8954	0.7227
STGCN [115]	22.6489	18.4551	0.9156	3.6042	2.7605	0.7316
STG2Seq [2]	18.0450	9.9415	0.8650	3.9843	2.4976	0.5152
Graph WaveNet [106]	13.0729	8.1037	0.9322	3.2943	1.9911	0.7003
MTGNN [105]	10.6842	5.6174	0.9550	2.7586	1.6347	0.8159
STSGCN [86]	10.0524	5.9302	0.9601	2.7973	1.7760	0.7937
AGCRN [3]	9.5107	5.3496	0.9659	2.8928	1.7793	0.7844
GMAN [127]	8.8448	5.2899	0.9695	2.6658	1.7044	0.8142
STTN [107]	9.1016	5.2710	0.9678	2.8485	1.7760	0.7871
Traffic-Transformer [9]	9.3527	5.4277	0.9656	2.7164	1.6715	0.8125
CCRNN [114]	9.5631	5.4979	0.9648	2.8382	1.7404	0.7934
ESG [112]	8.9063	5.0160	0.9696	2.7356	1.6342	0.8150
Ours	**8.2063**	**4.8872**	**0.9737**	**2.4852**	**1.5619**	**0.8422**

5.4.4 Implementation Details

Both datasets are aggregated into 30-minute interval with both historical and future horizons set to 12. This means predicting the future 6 hours of traffic demand based on the observations of the past 6 hours. Z-score normalization is applied to guarantee more stable training. The whole dataset is divided chronologically, with the last month's data equally partitioned into a validation and a test set.

The number of encoder/decoder layers is 3 for both datasets. For the spectral embedding function, the embedding length L for the X is 2 for both datasets, the embedding length M for the TE is set to 6 for the NYC-Bike and 10 for the NYC-Taxi dataset, respectively. For the parameters of Gaussian distribution in the matrix $\mathbf{A}$, μ and σ are set to be the mean and variance of each dataset. In the 2D-FFT, the number of blocks K is set to 8, the depth of the Fourier learning layer is 2, and the feed-forward network $f(\cdot)$ and $g(\cdot)$ in the time domain is designed as a two-layer MLP with ReLU as the activation function. For the spatial global optimization, the regularization parameter α is set to be 1 and 2 for the NYC-Bike and NYC-Taxi datasets, respectively. We adopt the Adam optimizer with a learning rate of 0.005 and 0.0015 for the NYC-Bike and NYC-Taxi datasets, respectively. All the hyperparameters are chosen based on the best performance on the validation set. The experiments are performed with PyTorch 1.11.0 on a Linux server equipped with NVIDIA Corporation TU102GL GPU.

5.4.5 Comparison with the State-of-the-Art

The prediction results averaged over 12 horizons are presented in Table 5.1. Compared to all the baselines, we can make the following observations: **(1)** Compared with the NYC-Taxi dataset, NYC-Bike has lower RMSE and MSE values, which is consistent with the lower demand for bikes compared to that of taxis in a metropolitan city like NYC. **(2)** The deep learning models outperform the statis-

tical methods in an overwhelming manner, which verifies their superiority in capturing the complex spatial-temporal correlations in traffic sequences. **(3)** GCN is the most frequently employed spatial learning module, but the GCN-based methods such as [2, 60, 105, 106, 115] obtain inferior performance because they struggle to explicitly capture the long-range correlations due to the local information propagation. Besides, Transformer-based methods such as [9, 107, 127] have a significant improvement by adopting the attention mechanism to enlarge the receptive field of correlation modeling. In contrast, our method can optimize the global spatial correlations in the traffic network, which outperforms both the local and pairwise models to promote forecasting performance further. **(4)** Our method is the only model to apply the frequency analysis along both the temporal and multivariate dimensions to leverage the rich spectral information and capture more intrinsic traffic patterns. **(5)** Our method consistently achieves the best performance on both datasets regarding all metrics.

Table 5.2 : Ablation Study of model components.

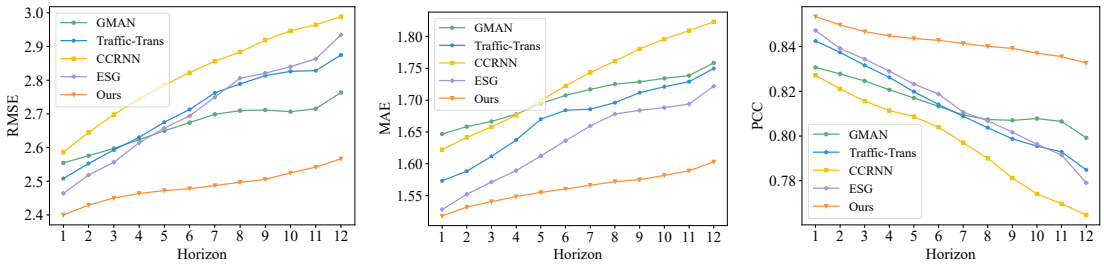
	Method		NYC-Taxi			NYC-Bike		
	Spatial	Temporal	RMSE	MAE	PCC	RMSE	MAE	PCC
A	Self-Attention	Self-Attention	8.7953	5.1593	0.9703	2.6175	1.6597	0.8215
B	Global Optimization	Self-Attention	8.2893	4.8546	0.9730	2.4942	1.5675	0.8412
C	Craph Convolution	Self-Attention	9.2718	5.2714	0.9665	2.6493	1.6927	0.8161
D	Self-Attention	1D-FFT	8.4717	4.9975	0.9721	2.5928	1.6166	0.8291
E	Self-Attention	2D-FFT	8.3610	4.9037	0.9728	2.5658	1.6054	0.8335
F	Self-Attention	Embedded 2D-FFT	8.2638	**4.7814**	0.9736	2.4955	1.5636	0.8409
G	Global Optimization	Embedded 2D-FFT	**8.2063**	4.8872	**0.9737**	**2.4852**	**1.5619**	**0.8422**

5.4.6 Ablation Study

We conduct detailed ablation studies on both datasets to demonstrate the effectiveness of each devised module in our method. To begin with, a simple baseline is constructed by adopting the self-attention mechanism to serve as both the spatial and temporal module for forecasting, which is denoted as 'A' in Table 5.2.

In this basic structure, the spatial correlations can only be captured by pairwise similarity computation, and temporal correlations can only be analyzed and learned in the time domain. Then, we replace the spatial attention with our global optimization module to generate the variant 'B'. The improved performance of 'B' on both datasets indicates the superiority of globally optimizing the spatial correlations in the citywide traffic network. Besides the pairwise matching mechanism, we also employ the local aggregation method GCN as the spatial learning module in 'C', and the severely degraded performance on both datasets indicates the insufficiency of its ability to capture the global correlations in the traffic network.

In the subsequent variants, we adopt spectral analysis as a substitute for temporal attention. We first apply 1D-FFT to only the temporal dimension of traffic inputs, which is represented as 'D' in Table 5.2. Then, a 2D-FFT is applied to both temporal and multivariate dimensions to generate the variant 'E'. We observe different extents of performance gain obtained on two datasets from introducing frequency domain learning, which is caused by distinctive traffic demand for bikes and taxis. This verifies the power of spectral analysis to help capture complex traffic patterns which are hard to obtain in the time domain. Besides, the performance is further improved by extending the spectral analysis from only the temporal dimension to the multivariate feature dimension, indicating the importance of interactions among multiple variates in traffic sequences. Furthermore, we incorporate the spectral embedding function into the 2D-FFT model, i.e., 'F'. This operation further



(a) RMSE (b) MAE (c) PCC

Figure 5.2 : Multi-horizon forecasting performance. Our method outperforms the baselines on all horizons (both short- and long-term) for all metrics.

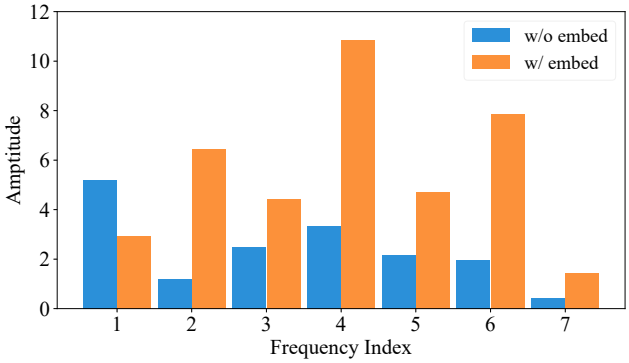
promotes forecasting performance, which demonstrates that enforcing the deep neural network to fit the high-frequency components in traffic data could capture more potentially useful traffic patterns. Finally, we integrate the embedded 2D-FFT and global optimization module into a unified framework 'G' to prove the complementary effect of each devised component, which achieves the best result on two of three metrics on the NYC-Taxi dataset and all metrics on the NYC-Bike dataset.

5.4.7 Multi-Horizon Forecasting Comparison

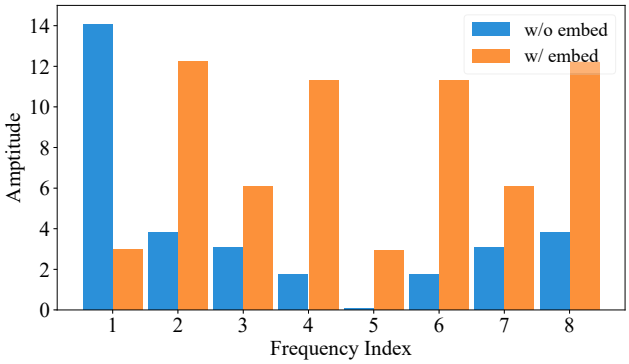
The comparison of multi-horizon forecasting results on the NYC-Bike dataset among four recently proposed state-of-the-art models and our method is presented in Fig. 5.2. We can see that our method outperforms other methods on three metrics over all 12 horizons, which verifies its superiority in both short-term (small horizon) and long-term (large horizon) forecasting tasks.

5.4.8 Frequency Component Visualization

We visualize the change of the frequency components reflected by the corresponding amplitude on the NYC-Bike dataset. Specifically, the 2D-FFT is conducted with/without applying the spectral embedding. Due to the conjugate symmetry



(a) Temporal dimension.



(b) Multivariate dimension.

Figure 5.3 : The Amplitude variations of high-frequency components.

of the FFT operation, there are only $(\frac{T}{2} + 1)$ frequency components preserved in the time dimension to avoid redundant frequencies (T is the length of the horizon in a batch of traffic data). From the visualization in Fig. 5.3, we can clearly observe that amplitudes of the high-frequency components are significantly promoted in terms of both temporal and multivariate dimensions by adopting the embedding function. This indicates the improved capability of deep neural networks to model the high-frequency variates of the input data.

5.4.9 The Global Optimization Visualization

We provide a visualization of the spatial correlation map learned by our global optimization module. From Fig. 5.4, we can observe a clear structural distribution

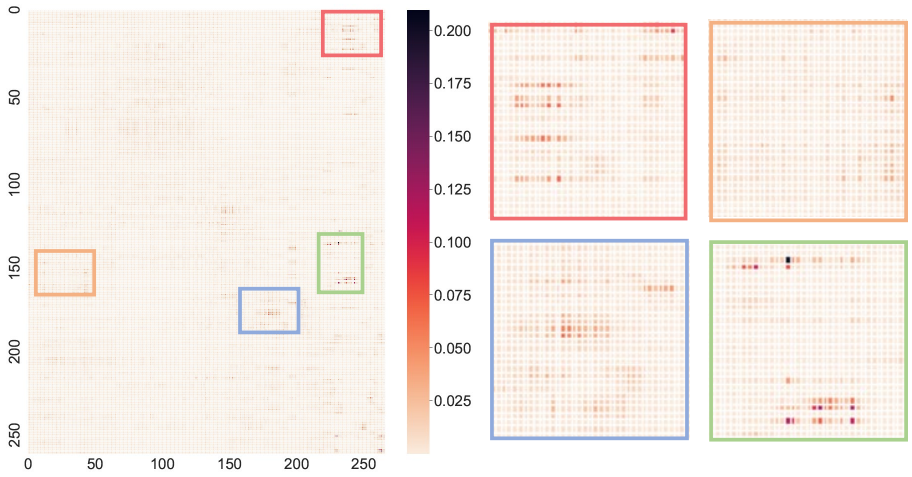


Figure 5.4 : The learned correlation map from the global optimization module.

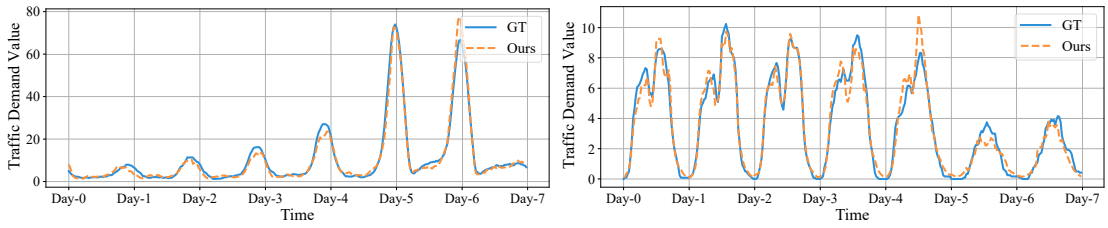
from our globally optimized correlation map, which indicates the importance of considering the overall traffic network in the traffic forecasting task. Besides, the unimportant correlations are discarded during the optimization to promote spatial feature learning for more accurate forecasting.

5.4.10 Prediction Visualization

We present the visualization of the predicted traffic sequences and ground truth over one week, with multiple horizons averaged to represent the whole forecasting result. In Fig. 5.5, the visualization is consistent with former quantitative results, which verifies that our method can achieve outstanding forecasting performance.

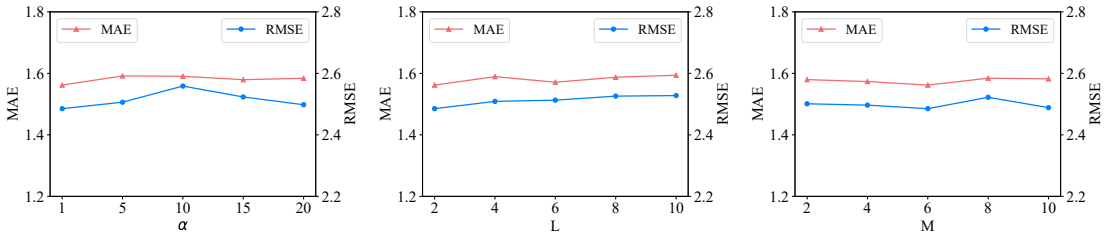
5.4.11 Parameter Analysis

We study the influence of three important parameters on the NYC-Bike dataset, i.e., the regularization parameter α (Eq. (5.16)) in the global spatial optimization, the embedding length L (Eq. (5.6)) and M (Eq. (5.8)) for X and TE respectively. From Fig. 5.6, we can observe that our method is stable under two evaluation metrics, with all parameters varying in a reasonable range.



(a) NYC-Taxi. (b) NYC-Bike.

Figure 5.5 : Comparison of the predicted traffic demand (Ours) and the ground truth (GT).



(a) Spatial regularization α (b) Embed length for X (c) Embed length for TE

Figure 5.6 : The influence of three parameters w.r.t. two metrics. When the parameters vary in a reasonable range, the performance of our method is stable.

5.5 Conclusion

In this chapter, we propose a novel traffic demand forecasting method to promote both temporal and spatial representation learning. For temporal learning, we introduce the embedded 2D spectral learning framework to explore the spectral correlations in the frequency domain. The framework comprises two well-devised modules: a spectral embedding function to explicitly extract both the low- and high-frequency components of the data, and a 2D spectral learning module to simultaneously explore the temporal and multivariate interactions in the frequency domain. For spatial learning, we design an optimal transport objective function to globally optimize the overall spatial correlations in the citywide traffic network

instead of the individual similarity calculation. Extensive experiments demonstrate the effectiveness of each designed module, and our method achieved state-of-the-art performance on two benchmarks.

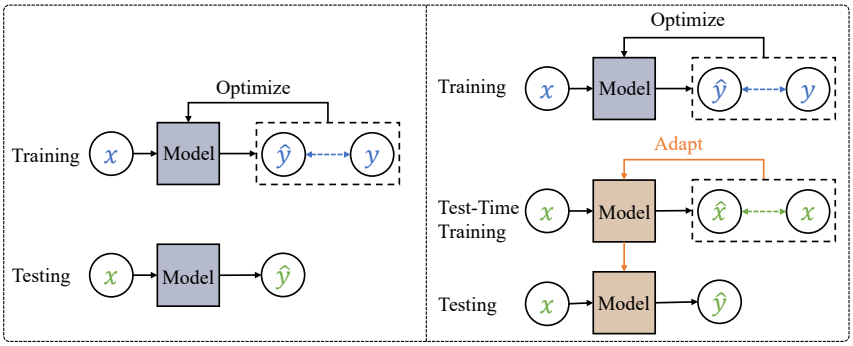
Chapter 6

Test-Time Training for Spatial-Temporal Forecasting

6.1 Introduction

Recent years have witnessed the success of deep neural networks in addressing the spatial-temporal forecasting task [60, 86, 95, 105, 112, 114, 127]. However, deep neural networks are notoriously vulnerable to the challenge of distribution shifts between training and test data [52]. This challenge becomes more severe for spatial-temporal forecasting, where training and testing data are divided *chronologically*. Specifically, the inherent temporal evolution makes the spatial-temporal data stream non-stationary, creating a natural gap between training and testing. Moreover, abrupt and unexpected factors will cause short-term data shifts that cannot be learned from the training data. For example, traffic accidents or meteorological conditions can cause rapid data shifts to traffic streams. As a result, these shifts can severely impede the trained forecasting model from generalizing to the unseen test data, leading to a noticeable performance drop.

To tackle the distribution shift, Test-Time Training (TTT) [87] has been proposed and gained increasing interest in different domains, such as image [49, 87, 96], video [25], and text [5]. Specifically, TTT adapts the trained model to the unknown distribution at test time by training auxiliary tasks (e.g., image rotation prediction) on *only one single test example without accessing the test label*. By design, TTT has the potential to address the test-time spatial-temporal shifts, but it is under-explored in spatial-temporal forecasting due to the lack of auxiliary tasks. In this chapter,



(a) Existing Methods (b) Our Method

Figure 6.1 : (a) Existing methods directly apply the trained model to test data, which may suffer from distribution shifts. (b) Our method proposes a new test-time training framework, which can effectively adapt the trained model to test data before testing.

we propose a novel test-time training framework to tackle the non-stationary shifts in time-evolving spatial-temporal data. The framework is designed to learn directly from the test data, enabling it to adapt to unknown data shifts before making the final prediction. As shown in Figure 6.1, our design can effectively break the limitation of existing methods, whose models remain fixed after the training stage. In contrast, our model can further adapt the trained model to test data.

Unlike other domains, such as image and text, which have only one example for test-time training, the inherent data structure of the spatial-temporal task is well-suited for test-time training. Specifically, at the test stage, spatial-temporal forecasting simultaneously predicts the values of future time steps for all geographical locations (several hundred), providing an effective batch size for test-time training. This way, the spatial-temporal distribution of test data is explored for accurate forecasting. Furthermore, the effective batch size mitigates the overfitting risk of test-time adaptation.

Since the test-time training should not access the target label, it cannot be read-

ily applied to a supervised forecasting model. This presents a significant hurdle when attempting to implement adaptation during test time. In response to this issue, we introduce a novel bidirectional cycle-consistent architecture for spatial-temporal forecasting, enabling the implementation of the test-time training strategy. The architecture consists of *a forward and a backward cyclic network*, each of which encapsulates the supervised and self-supervised learning to conduct prediction and reconstruction tasks in both directions (forward and backward) simultaneously. Specifically, a shared encoder is designed for both networks to extract the spatial-temporal correlations of the input streams. Two direction-aware decoders are proposed to conduct forward prediction and backward recollection tasks separately, where supervised losses are applied. By carefully assembling the modules, two auxiliary tasks are conducted: *forward*→*backward* reconstruction and *backward*→*forward* reconstruction, where self-supervised losses are applied. During training, both supervised and self-supervised losses are utilized. However, at the test-time training stage, only the self-supervised losses are used to adapt the trained model without accessing the target labels.

Besides benefiting the test-time training stage, this bi-cyclic structure can also bring advantages for our main forecasting task at the training stage. During training, the shared encoder of the forward and backward cyclic networks is optimized by four tasks: the supervised prediction, recollection tasks, and two self-supervised cyclic reconstruction tasks. All four tasks are intertwined to learn better spatial-temporal encoding by enforcing consistency among the cyclic networks. Similarly, the forward decoder enjoys joint training of four tasks to promote the final forecasting performance. The two self-supervised tasks co-exist at both the training and test stages, bridging the model gap between training and test-time training.

6.2 Related Work

Test Time Adaptation Domain adaptation [39, 65] is one of the promising paradigms to tackle the distribution shift, which aims to learn a cross-domain representation by employing the unlabeled data from the target domain besides the training data from the source domain. However, it can only anticipate the specific distributions where the data is prepared in advance. This pitfall has motivated the emerging test-time adaptation to adapt the trained model to unseen domains on the fly, without accessing the training data or the labels of target data. It has been successfully applied to various domains, such as image classification [87, 96], reinforcement learning [36], natural language processing [5] and video tracking [25]. Typically, test-time training (TTT) [87] adapts the trained model for each individual test sample via self-supervised learning. During the training, the model is optimized by both the main task (e.g., image classification) and the proxy task (e.g., image rotation prediction). Then, at the test stage, the model is further adapted to each single test example with the proxy task to reflect the test distribution.

Following this, several works are proposed under different assumptions. For example, TENT [96] deals with the source-free domain adaptation where only the model and target data are available. It discards the model's training on the source data and proposes to adapt the model during test time by minimizing the entropy of the prediction outputs. [61] exploits the information maximization and self-supervised pseudo-labeling to align the representations from the target domains to the source domain. TTT++ [64] is a modified version of TTT that proposes a feature alignment strategy to encourage the closer relationship of the training and testing feature distributions with access to an entire dataset. [28] successfully applies masked autoencoders (MAE) to the test-time training framework and achieves outstanding performance on the object recognition task.

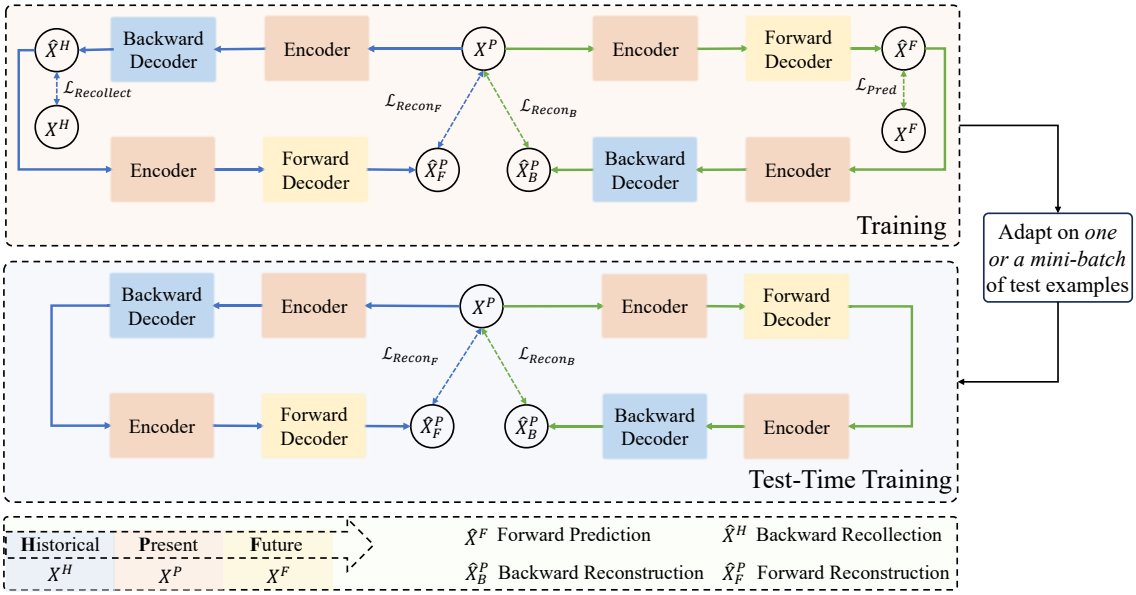


Figure 6.2 : The proposed bidirectional cycle-consistent architecture. Each network consists of a shared encoder and two direction-aware decoders. During training, we jointly optimize the prediction ($\mathcal{L}_{Pred}$), recollection $\mathcal{L}_{Recollect}$ and reconstruction ($\mathcal{L}_{Recon_F}, \mathcal{L}_{Recon_B}$) tasks from different directions. During the test-time training, the model is further adapted to the test data by minimizing the reconstruction losses ($\mathcal{L}_{Recon_F}, \mathcal{L}_{Recon_B}$) on one example or a mini-batch, which can deal with the shifted distribution before making the final prediction.

Despite their effectiveness, the test-time adaptation to the spatial-temporal data has never been explored. In this chapter, we design a test-time training framework equipped with a bidirectional cycle-consistent structure, which employs self-supervised reconstruction tasks to adapt the model to unseen test data shifts before making the final forecasting result.

6.3 Method

6.3.1 Problem Formulation

This chapter aims at multi-horizon spatial-temporal forecasting, which predicts future spatial-temporal conditions of multiple time steps based on the observed data.

Specifically, the spatial distribution is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ is the set of spatial nodes, $|\mathcal{V}| = N$ is the cardinality, $\mathcal{E}$ is the edge reflecting the spatial correlations. We construct the spatial embedding from this graph and temporal embedding from the specific hourly, daily and weekly temporal features of the data to represent the spatial-temporal embedding (STE), which is concatenated to the input before fed into the model.

To construct the bidirectional cycle-consistent structure, we divide the spatial-temporal data into three consecutive segments chronologically. Specifically, we adopt the data from both the historical and present time segments represented by $X^H \in \mathbb{R}^{H\times N\times d}$, $X^P \in \mathbb{R}^{P\times N\times d}$, where the H and P stands for the horizon of the historical and present time, d is the feature dimension of the spatial-temporal data. The aim is to predict the data $X^F \in \mathbb{R}^{F\times N\times d}$ in the future F horizons based on both the input data from the historical and present time and the spatial-temporal embeddings from all times. Therefore, we train a model f parameterized by parameters θ to predict an estimation of X^F:

$$\hat{X}^F = f_\theta(X^H, X^P)\,. \tag{6.1}$$

6.3.2 Preliminary

Spatial-Temporal Attention

To dynamically capture the spatial-temporal correlations within data, we employ spatial-temporal attention as the fundamental module for our shared encoder and direction-aware decoders. Specifically, Spatial Attention and Temporal Attention

perform multi-head attention along the spatial and temporal dimensions in data, respectively. Typically, the attention is calculated as the scaled dot-product:

$$\text{Attention}(Q, K, V) = \text{Softmax}\big(\frac{QK^T}{\sqrt{D}}\big)V\,, \tag{6.2}$$

where Q, K and V respectively denote the query, key and value of input data, and D is the hidden dimension of input.

Normally, the Multi-head self-attention is applied to extract features from different subspaces of input data:

$$\text{MHA}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_k)W_O\,, \tag{6.3}$$

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V)\,, \tag{6.4}$$

where k is the number of heads , $W_i^Q \in \mathbb{R}^{D\times D/k}$, $W_i^K \in \mathbb{R}^{D\times D/k}$, $W_i^V \in \mathbb{R}^{D\times D/k}$ and $W_O \in \mathbb{R}^{D\times D}$ are the weight parameters.

Besides the self-attention, we also adopt the cross-attention between each encoder and decoder to build the connection between two consecutive temporal segments, which takes the spatial-temporal embeddings from corresponding time as query and key, and the output of the encoder as value to conduct the multi-head attention.

6.3.3 Bidirectional Cycle-Consistent Structure

In this chapter, we design a bidirectional cyclic spatial-temporal network, which effectively leverages the encoder and decoders to establish learning cycles in both the forward and backward directions. The architecture of the proposed cycle-consistent network is shown in Figure 6.2.

To make the model direction-aware, we design a bidirectional encoder-decoder structure based on the three consecutive temporal segments in chronological order. Specifically, the encoder is shared in both directions to extract the spatial-temporal patterns from the input. Benefitting from the cross-attention to establish the correlations between two different segments, the forward decoder could learn to predict

from the present to the future, while the backward decoder can learn to recollect from the present to the past. This pictorially is like a 'T'-structure to have the shared encoder at the bottom and two branches as forward and backward decoders, respectively:

$$\hat{X}^F = Dec_F(Enc(X^P)), \tag{6.5}$$

$$\hat{X}^H = Dec_B(Enc(X^P)), \tag{6.6}$$

where $Enc(\cdot)$, $Dec_F(\cdot)$ and $Dec_B(\cdot)$ denote the shared encoder and forward/backward decoder (with cross-attention included), respectively.

For these two tasks, the objective is to minimize the mean absolute error between the prediction output and the future ground truth, as well as the recollection result and historical ground truth:

$$\mathcal{L}_{Pred} = \|\hat{X}^F - X^F\|, \tag{6.7}$$

$$\mathcal{L}_{Recollect} = \|\hat{X}^H - X^H\|, \tag{6.8}$$

where X^F and X^H are the labels for the future and historical data.

However, in this 'T'-shaped structure, the outputs of both decoders are independent of each other because the forward and backward decoders are not optimized for a consistent target. Inspired by the successful application of cycle-consistency in the community of visual correspondence learning [41, 89, 99], we further design a bidirectional-cyclic structure by reconstructing the input data through a cyclic learning manner from two directions. Specifically, in the forward cycle, the encoder and the backward decoder are applied to the predicted output to reconstruct the input from what the forward decoder predicted. Similarly, for the backward direction, the encoder and forward decoder are also applied to reconstruct the same input

based on what the backward decoder recollected from the past:

$$\hat{X}_F^P = Dec_F(Enc(\hat{X}^H)) \,, \tag{6.9}$$

$$\hat{X}_B^P = Dec_B(Enc(\hat{X}^F)) \,. \tag{6.10}$$

The objective is then designed for the model to reconstruct the input by minimizing the mean absolute error between the reconstruction and input in both forward and backward directions:

$$\mathcal{L}_{Recon_F} = \|\hat{X}_F^P - X^P\| \,, \tag{6.11}$$

$$\mathcal{L}_{Recon_B} = \|\hat{X}_B^P - X^P\| \,. \tag{6.12}$$

In this bidirectional-cyclic-consistent structure, the forward and backward networks encapsulating both the supervised forecasting and self-supervised reconstruction tasks are jointly optimized by minimizing the total loss as follows:

$$\mathcal{L}_{train} = \mathcal{L}_{Pred} + \alpha(\mathcal{L}_{Recollect} + \mathcal{L}_{Recon_F} + \mathcal{L}_{Recon_B}) \,.$$

As the main task of our method is to predict the future, the losses from other tasks are regarded as the regularization with α being the regularization parameter.

The forward decoder is optimized to generate accurate predictions in the forward cycle, while it is also optimized to generate the reconstruction of the input data in the backward cycle. Direct access to the label in each decoder can further benefit the other decoder by providing more informative input for the reconstruction task. In this cyclic way, the targets of the forward and backward networks are intertwined with each other. Thus, the shared encoder and forward decoder can be jointly optimized for better forecasting performance.

6.3.4 Test-Time Training for Spatial-Temporal Forecasting

For existing methods, the model optimized on the training stage is fixed when conducting the forecasting task on test data, which makes it hard to generalize the

possible distribution shift widely existing in the evolving spatial-temporal data.

To deal with this challenge, we propose a test-time training strategy for the spatial-temporal forecasting task, which dynamically adapts the model learned from the training stage to fit each unseen test data in a self-supervised manner. Our proposal is motivated by the increasing popularity of test-time adaptation methods with wide application in image classification tasks [28, 64, 87, 96]. Different from image data with an independent input and discrete label space, which can only provide one example during the test-time training, our spatial-temporal data share the temporally continuous space for the input and output, which has strong spatial-temporal correlations among the consecutive time steps and spatial nodes. This special data structure can provide an effective batch size for test-time training to capture the spatial-temporal distribution without overfitting to a single time stream.

However, employing the test data without labels requires self-supervised learning regarding our spatial-temporal forecasting task to support the model adaptation in the test time. Rather than having to introduce the extra proxy task, such as predicting the rotation angle of the image [87] or minimizing the entropy of the output [96], we can benefit directly from our bidirectional cycle-consistent network. The forward reconstructing from the recollected historical output as well as the backward reconstructing from the predicted output, which both use the original input as the supervision, can be directly employed as the self-supervised task for the test-time training.

Specifically, we first load the model from the end of the training and get the reconstruction of the input data from both directions as follows:

$$\hat{X}_F^P = Dec_F(Enc(Dec_B(Enc(X^P)))) \,, \tag{6.13}$$

$$\hat{X}_B^P = Dec_B(Enc(Dec_F(Enc(X^P)))) \,. \tag{6.14}$$

Then the model is optimized by minimizing the reconstruction error from the

two cyclic networks weighted by a regularization parameter β:

$$\mathcal{L}_{TTT} = \|\hat{X}_F^P - X^P\| + \beta\|\hat{X}_B^P - X^P\| . \tag{6.15}$$

During the test-time optimization, the parameters of the model are adjusted exclusively for only one or a mini-batch of test examples before making the final forecasting on it (them). In this way, the capability limitations of the fixed model are broken by dynamically adapting to each test data according to the specific spatial-temporal distributions with potential shifts.

Our model can also be easily applied to the online learning setting, where the model is continuously updated without re-initializing the original parameters for each test data.

6.4 Experiments

6.4.1 Datasets

To verify the effectiveness of our model in spatial-temporal forecasting task, we conduct experiments on two real-world traffic datasets collected by NYC OpendData*. The two datasets represent the traffic demand in terms of taxi and bike respectively by recording the pick-up and drop-off order information in NYC. The detailed information is presented as follows:

- NYC-Bike: It collects daily user orders for sharing bikes at 250 stations in NYC. The time spans from April 1st, 2016 to June 30th, 2016.
- NYC-Taxi: It contains taxi demand data collected from 266 traffic stations with time spanning from April 1st, 2016 to June 30th, 2016.

*https://opendata.cityofnewyork.us/

6.4.2 Baselines

We compare our method with several baselines that can be divided into two categories: statistical models and deep learning-based models. The first category includes XGBoost [13]. The second category consists of DCRNN [60], STGCN [115], STG2Seq [2], STSGCN [86], MTGNN [105], Traffic Transformer [9], GMAN [127], CCRNN [114], GTS [82] and ESG [112].

6.4.3 Evaluation Metrics

We adopt three widely-used metrics in spatial-temporal forecasting tasks to evaluate the performance of our model compared with all baselines, i.e., Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Empirical Correlation Coefficient (CORR) [112]. The first two metrics measure the difference between the prediction and the label; the lower, the better. The last metric measures the linear correlation between the predicted and ground truth sequences; the larger, the better.

The detailed information on baselines and metrics can be found in the Supplementary.

6.4.4 Implementation Details

Both datasets are aggregated by 30-minute intervals, and all the horizons for historical (H), present (P) and future (F) are set to 12, which forms three 6-hour temporal segments. Z-score normalization is applied to the input data for more stable training. The entire dataset is chronologically divided, with the data from the last month evenly split into both a validation and a test set.

For the attention module in the encoder and decoders, the number of the head k is set to 8, and the hidden dimension D is 64. The stacking layers for the shared encoder, forward and backward decoders are set to 3. During the training stage, we

Table 6.1 : Comparison with the state-of-the-art methods on two datasets w.r.t. three metrics across various horizon settings (the best performance is highlighted in bold).

Dataset	Method	Horizon 3			Horizon 6			Horizon 12			All		
		RMSE	MAE	CORR	RMSE	MAE	CORR	RMSE	MAE	CORR	RMSE	MAE	CORR
NYC-Bike	XGBoost	3.7048	2.2167	0.5232	4.1747	2.5511	0.3614	4.3925	2.7091	0.2894	4.0494	2.4689	0.4107
	DCRNN	3.0172	1.7917	0.6967	3.2369	1.9078	0.6609	3.5100	2.0325	0.6196	3.2274	1.8973	0.6601
	STGCN	2.6256	1.6456	0.7539	3.8368	2.2827	0.6282	4.3713	2.6052	0.4521	3.7829	2.2076	0.5933
	STG2Seq	3.4669	2.0409	0.5999	3.9145	2.2630	0.5079	4.2373	2.5163	0.4443	3.7843	2.2055	0.5413
	STSGCN	2.7328	1.6973	0.7386	2.8861	1.7416	0.7179	3.0548	1.8224	0.6903	2.8846	1.7538	0.7126
	MTGNN	2.5962	1.5668	0.7626	2.7588	1.6525	0.7447	3.3068	1.7892	0.6931	2.7791	1.6595	0.7353
	Traffic-Trans	2.5931	1.6116	0.7579	2.7131	1.6841	0.7391	2.8750	1.7496	0.7071	2.7140	1.6715	0.7374
	GMAN	2.5974	1.6665	0.7432	2.6744	1.7073	0.7297	2.7635	1.7584	0.7142	2.6658	1.7044	0.7302
	CCRNN	2.6538	1.6565	0.7534	2.7561	1.7061	0.7411	2.9436	1.8040	0.7029	2.7674	1.7133	0.7333
	GTS	2.7628	1.7159	0.7248	2.9287	1.7769	0.7007	3.1649	1.8905	0.6622	2.9258	1.7798	0.6985
	ESG	2.5529	1.5483	0.7638	2.6484	1.6026	0.7511	2.8778	1.7173	0.7152	2.6727	1.6129	0.7449
	Ours	**2.3734**	**1.5199**	**0.7783**	**2.4483**	**1.5498**	**0.7687**	**2.5282**	**1.5896**	**0.7589**	**2.4476**	**1.5511**	**0.7690**
NYC-Taxi	XGBoost	15.0372	8.4121	0.6862	21.3395	11.8491	0.4433	26.7073	15.7165	0.0452	21.1994	11.6806	0.4416
	DCRNN	12.3223	7.0655	0.7591	15.1599	8.6639	0.6634	17.8194	10.5095	0.5395	14.8318	8.4835	0.6671
	STGCN	11.2175	6.1441	0.8090	14.0360	7.6797	0.7470	18.7168	10.2211	0.5922	14.6473	7.8435	0.7257
	STG2Seq	14.0756	7.7274	0.7258	19.1757	10.5066	0.5429	24.5691	14.3603	0.2855	19.2077	10.4925	0.5389
	STSGCN	10.5381	5.6448	0.8370	10.8444	5.7634	0.8302	11.9443	6.3185	0.7988	10.9692	5.8299	0.8242
	MTGNN	10.3394	5.6775	0.8374	10.7534	5.8168	0.8312	12.5164	6.5285	0.7972	10.9472	5.9192	0.8249
	Traffic-Trans	8.5293	5.0112	0.8587	9.0698	5.2171	0.8518	9.9286	5.6392	0.8319	9.0040	5.2098	0.8501
	GMAN	8.4650	5.0852	0.8455	8.8780	5.2776	0.8331	9.3772	5.6314	0.8128	8.8448	5.2899	0.8325
	CCRNN	9.1983	5.3573	0.8575	9.4547	5.4093	0.8509	10.2733	5.7789	0.8380	9.4849	5.4396	0.8506
	GTS	10.7796	6.2337	0.7974	13.0215	7.3251	0.7299	14.9906	8.5328	0.6524	12.7511	7.2095	0.7348
	ESG	8.5745	4.8750	0.8656	9.0125	5.0500	0.8592	9.7857	5.4019	0.8450	8.9759	5.0344	0.8592
	Ours	**7.3605**	**4.5073**	**0.8746**	**8.0176**	**4.7217**	**0.8657**	**8.6849**	**5.0690**	**0.8527**	**7.9896**	**4.7368**	**0.8652**

adopt the Adam optimizer with a learning rate of 0.005 and 0.0015 for the NYC-Bike and NYC-Taxi datasets, respectively, and the batch size is 16 for both datasets. All the hyperparameters are chosen based on the best performance on the validation set. The regularization parameter α is set to be 0.01 for both datasets. During the test-time training, the learning rate has to be set to a smaller value to alleviate the overfitting. Thus, we set it to be 7e-5 for the NYC-Taxi and 4e-5 for the NYC-Bike dataset. In this stage, only the reconstruction losses (Eq. (6.15)) are employed to adapt the model to test data without the target labels, and the regularization

parameter β is set to 4.0 and 1.0 respectively for NYC-Taxi and NYC-Bike. Different from the image classification task, where the aim of the self-supervised task differs from that of the main task, benefitting from our bi-cyclic structure, our model can enjoy minibatch training during the test time, and the batch size is set to 8 in both datasets for better performance. Noticeably, for the first batch of data, we can only access the first sample to avoid the information leak in the future. Thus, the other seven samples are from the validation set, which can be achieved by the cached data in the real-world application. For the online version, the learning rate is set to 1e-5 and 2e-5 for the NYC-Taxi and BYC-Bike, and the regularization parameter β is 1.0 for both datasets.

6.4.5 Results and Analysis

The performance of all methods at the specific horizon and the average result over all 12 horizons are presented in Table 6.1, from which we can make the following observations:

- The deep learning-based spatial-temporal forecasting models consistently outperform their statistical counterparts by a large margin, which is attributed to their capability to capture the hierarchical features and complex spatial-temporal correlations in traffic data.

- Among these baselines, our test-time training spatial-temporal forecasting model is the only method to deal with the distribution shift in spatial-temporal data by learning directly from the test data. In contrast, other models are fixed at the end of training before conducting forecasting, which constrains their forecasting performance because of the lack of capability to generalize to the shifted spatial-temporal distributions in test data.

- Our model achieves the best performance on both datasets over all the metrics.

Table 6.2 : Ablation Study of Model Components.

Variant	NYC-Taxi			NYC-Bike		
	RMSE	MAE	CORR	RMSE	MAE	CORR
Base	8.8448	5.2899	0.8325	2.6658	1.7044	0.7302
+Bi-Cycle	8.4238	4.8941	0.8596	2.5235	1.5889	0.7631
+TTT (B-1)	8.2258	4.8037	0.8638	2.4752	1.5573	0.7672
Online Learning	8.0996	4.7674	0.8654	2.4605	1.5583	0.7680
+TTT (B-8)	7.9896	4.7368	0.8652	2.4476	1.5511	0.7690

This demonstrates the effectiveness of our test-time training spatial-temporal forecasting model equipped with the bidirectional cycle-consistent structure.

6.4.6 Ablation Study

We conduct detailed ablation studies on both datasets to demonstrate the effectiveness of the bidirectional cycle-consistent structure and test-time training strategy in our spatial-temporal forecasting model. Firstly, a simple baseline is constructed by adopting the self-attention-based encoder-decoder architecture, which is optimized only on the training data through the supervised forward forecasting task. We denote it as 'Base' in Table 6.2.

Then, we extend the base model by constructing the forward and backward learning cycle with the shared encoder and bi-directional decoders, which can simultaneously conduct the predicting (recollecting) and reconstructing tasks from both directions. We denote this setting as '+Bi-Cycle' in Table 6.2. After joint learning of all tasks, the performance can achieve obvious promotion over the basic single-direction learning. By encouraging consistency among the two cyclic networks, the shared encoder and the forward decoder are optimized in a cyclic manner back and forth, which significantly consolidates their capability to generate a more accurate

Table 6.3 : Variants of Bi-Cyclic Structure.

Variant	NYC-Taxi			NYC-Bike		
	RMSE	MAE	CORR	RMSE	MAE	CORR
w/ all tasks	8.4238	4.8941	0.8596	2.5235	1.5889	0.7631
w/o recollection	8.6890	5.0502	0.8567	2.6126	1.5941	0.7575
w/o forward recons.	8.5098	4.9703	0.8545	2.6165	1.6152	0.7512
w/o backward recons.	8.5546	4.9592	0.8573	2.5586	1.5989	0.7567

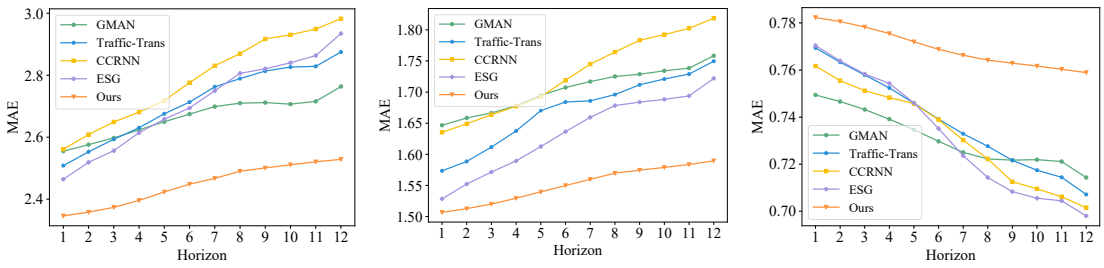
and reliable representation for future forecasting.

Lastly, we employ the reconstruction error (Eq. (6.15)) as the self-supervision to adapt the model to the test data without accessing their labels, which is referred to as the '+TTT' (with different batch sizes) in Table 6.2. The results show that this strategy leads to further improvements over the fixed model solely optimized during the training stage. This corroborates the importance of test-time training. By adjusting its parameters according to the specific distribution of different test samples, the model can learn to fit the unexpected spatial-temporal shift in the unseen data, which can directly extend the capability of forecasting test data. Moreover, our model can enjoy the mini-batch strategy during the test-time training. By feeding a reasonable amount of data to the model, the problem of overfitting can also be alleviated, which is also validated by the improvement from using batch size as 1 ('B-1') to batch size as 8 ('B-8').

Our model can also be easily scaled to the online learning setting, where the model is continuously optimized by each sequentially coming data. From the table, we can see that our method also achieves outstanding forecasting performance under continuous learning.

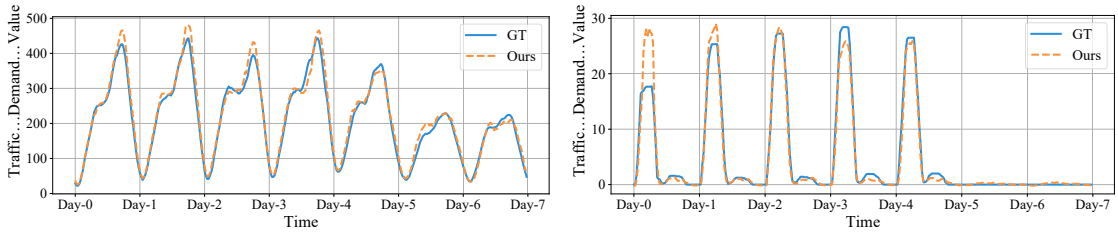
We also verify the importance of each different task in the bi-cyclic structure in Table 6.3 on both datasets, which is implemented without test-time training. As

seen from the results, the absence of each loss from the corresponding task can lead to a performance drop to a different extent, which indicates the critical function of each component in the bi-cyclic network. This can be explained by the intertwined target of each task, where the joint optimization of all forward and backward learning can consistently promote the representation learning for the final forecasting.



(a) RMSE (b) MAE (c) CORR

Figure 6.3 : Multi-horizon forecasting performance on NYC-Bike in terms of three metrics. Our method outperforms all state-of-the-art on all horizons.



(a) NYC-Taxi. (b) NYC-Bike.

Figure 6.4 : Comparison between the predictions by our method (Ours) and the ground truth values (GT).

6.4.7 Multi-horizon Forecasting

In Figure 6.3, we provide the multi-horizon forecasting comparison on the NYC-Bike dataset between four recently proposed state-of-the-art models and our method. We can see that our model exhibits a consistent superiority to all other methods

on three metrics over all horizons, which verifies its powerful capability in both short-term (small horizon) and long-term (large horizon) forecasting.

6.4.8 Visualization Results

In this subsection, we visualize the predicted traffic sequence and ground truth over one week on both datasets, with multiple horizons averaged to represent the forecasting result. The visualization in Figure 6.4 is consistent with quantitative results in Table 6.1, which verifies that our method can achieve outstanding forecasting performance.

6.5 Conclusion

In this chapter, we propose a test-time training strategy for the spatial-temporal forecasting task, which is incorporated into a well-designed bidirectional cycle-consistent structure. Specifically, benefiting from the test-time training design, our model can learn from different test examples to adapt itself to the unknown spatial-temporal distributions shifted from the training data. By designing the bi-cyclic structure, the test-time training for the spatial-temporal data can be implemented by directly employing the reconstruction tasks as self-supervision. The joint optimization of multiple tasks for shared modules can also benefit the spatial-temporal representation capability for the final forecasting task. The extensive experiments on the spatial-temporal traffic forecasting task demonstrate the effectiveness of each component in our model.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis targets traffic forecasting tasks by enhancing spatial-temporal representation learning to capture the complicated dynamic spatial-temporal correlations in traffic data. We propose four different spatial-temporal representation learning models in correspondence to the four challenges that are under-explored in the current traffic forecasting applications. Firstly, motivated by the imbalanced complexity of traffic forecasting tasks in terms of spatial and temporal distribution, we propose a Dynamic Halting Mechanism for the Transformer to balance the computation resource for the tasks according to the diverse complexity, which ensures the efficiency and the performance of the traffic flow forecasting. Secondly, we aim to address risk sample scarcity and imbalance challenges in the traffic accident datasets. We design a novel multi-kernel CNN structure to assign diverse receptive fields for regions with different spatial granularities to capture the hierarchical spatial representation. For the temporal representation, we design a novel contrastive learning approach with a mixup strategy for risk sample augmentation to contrastively learn the risk representation. In the third model, we build a frequency domain Transformer, which can conduct spatial-temporal representation learning from both the time and spectral domain. Specifically, we propose 2D spectral learning to solve the spectral bias challenge in deep neural networks. Finally, we propose a test-time training for spatial-temporal representation learning, which can effectively alleviate the distribution shift between the training and testing data in the traffic forecast-

ing tasks. All the methods in this thesis have achieved state-of-the-art forecasting performance, which is demonstrated through extensive experiments conducted on various real-world traffic datasets.

7.2 Future Work

Large languagemodels (LLMs) and foundational models have recently gained explosive surge in popularity and are witnessing rapid development across a wide range of domains, such as natural language processing (NLP) [21, 76, 77], computer vision (CV) [6,55,90], and other research communities [67,85]. The core of LLMs lies in the pre-training of a large language model from a large number of text corpora and billions of model parameters to facilitate the fine-tuning of the downstream tasks. One significant advantage of LLMs is that, in contrast to the conventional deep learning models, which need to be designed specifically for different tasks and data, they can be applied to a diverse range of tasks with a unified framework.

Although having achieved remarkable success, the applications of LLMs to spatial-temporal representation learning are still limited. The only attempt at applying the pre-trained LLM to traffic forecasting is [109], which generates prompts by converting the numerical temporal sequences into natural language sentences. This approach allows the direct processing of prompts by pre-trained LLMs for downstream forecasting tasks, providing insightful guidance on the application of pre-trained language models. However, there is a need for further exploration in future work to address how to generate more informative prompts for LLMs to enhance their effectiveness in downstream tasks. Additionally, the exploration of techniques to leverage current LLMs for capturing spatial correlations remains an unexplored avenue.

Bibliography

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *stat*, vol. 1050, p. 21, 2016.

[2] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, "Stg2seq: spatial-temporal graph to sequence model for multi-step passenger demand forecasting," in *28th International Joint Conference on Artificial Intelligence, IJCAI 2019*. International Joint Conferences on Artificial Intelligence, 2019, pp. 1981–1987.

[3] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *Advances in neural information processing systems*, vol. 33, pp. 17 804–17 815, 2020.

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

[5] P. Banerjee, T. Gokhale, and C. Baral, "Self-supervised test-time learning for reading comprehension," *arXiv preprint arXiv:2103.11263*, 2021.

[6] H. Bao, L. Dong, S. Piao, and F. Wei, "Beit: Bert pre-training of image transformers," *arXiv preprint arXiv:2106.08254*, 2021.

[7] R. Basri, M. Galun, A. Geifman, D. Jacobs, Y. Kasten, and S. Kritchman, "Frequency bias in neural networks for input of non-uniform density," in *International Conference on Machine Learning*. PMLR, 2020, pp. 685–694.

[8] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv preprint arXiv:1703.04691*, 2017.

[9] L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu, "Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting," *Transactions in GIS*, vol. 24, no. 3, pp. 736–755, 2020.

[10] C. Chen, X. Fan, C. Zheng, L. Xiao, M. Cheng, and C. Wang, "Sdcae: Stack denoising convolutional autoencoder model for accident risk prediction via traffic big data," in *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*. IEEE, 2018, pp. 328–333.

[11] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia, "Freeway performance measurement system: mining loop detector data," *Transportation Research Record*, vol. 1748, no. 1, pp. 96–102, 2001.

[12] K. Chen, G. Chen, D. Xu, L. Zhang, Y. Huang, and A. Knoll, "Nast: Non-autoregressive spatial-temporal transformer for time series forecasting," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1684–1694.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[15] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern*

Recognition, 2021, pp. 15 750–15 758.

[16] X. Chen, S. Chen, J. Yao, H. Zheng, Y. Zhang, and I. W. Tsang, "Learning on attribute-missing graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[17] Y. Chen, I. Segovia, and Y. R. Gel, "Z-gcnets: Time zigzags at graph convolutional networks for time series forecasting," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1684–1694.

[18] L. Chi, B. Jiang, and Y. Mu, "Fast fourier convolution," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4479–4488, 2020.

[19] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, D. Belanger, L. Colwell *et al.*, "Masked language modeling for proteins via linearly scalable long-context transformers," *arXiv preprint arXiv:2006.03555*, 2020.

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

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

[22] C. M. S. Distances, "Lightspeed computation of optimal transport," *Advances in Neural Information Processing Systems*, vol. 26, pp. 2292–2300, 2013.

[23] W. Duan, X. He, Z. Zhou, L. Thiele, and H. Rao, "Localised adaptive spatial-temporal graph neural network," *arXiv preprint arXiv:2306.06930*, 2023.

[24] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang, "Multi-horizon time series forecasting with temporal attention learning," in *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining*, 2019, pp. 2527–2535.

[25] Y. Fu, S. Liu, U. Iqbal, S. De Mello, H. Shi, and J. Kautz, "Learning to track instances without video annotations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8680–8689.

[26] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[27] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[28] Y. Gandelsman, Y. Sun, X. Chen, and A. Efros, "Test-time training with masked autoencoders," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29 374–29 385, 2022.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[30] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[31] J. Gu, Q. Zhou, J. Yang, Y. Liu, F. Zhuang, Y. Zhao, and H. Xiong, "Exploiting interpretable patterns for flow prediction in dockless bike sharing systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 2, pp. 640–652, 2020.

[32] J. Guibas, M. Mardani, Z. Li, A. Tao, A. Anandkumar, and B. Catanzaro, "Adaptive fourier neural operators: Efficient token mixers for transformers," *arXiv preprint arXiv:2111.13587*, 2021.

[33] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[34] S. Guo, Y. Lin, H. Wan, X. Li, and G. Cong, "Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 11, pp. 5415–5428, 2021.

[35] M. Gupta, H. Kodamana, and S. Ranu, "Frigate: Frugal spatio-temporal forecasting on road networks," *arXiv preprint arXiv:2306.08277*, 2023.

[36] N. Hansen, R. Jangir, Y. Sun, G. Alenyà, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang, "Self-supervised policy adaptation during deployment," *arXiv preprint arXiv:2007.04309*, 2020.

[37] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*. Pmlr, 2018, pp. 1989–1998.

[40] C. Huang, C. Zhang, P. Dai, and L. Bo, "Deep dynamic fusion network for traffic accident forecasting," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2673–2681.

[41] A. Jabri, A. Owens, and A. Efros, "Space-time correspondence as a contrastive random walk," *Advances in neural information processing systems*, vol. 33, pp. 19 545–19 560, 2020.

[42] J. Ji, J. Wang, C. Huang, J. Wu, B. Xu, Z. Wu, J. Zhang, and Y. Zheng, "Spatio-temporal self-supervised learning for traffic flow prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4356–4364.

[43] J. Jiang, C. Han, W. X. Zhao, and J. Wang, "Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction," *arXiv preprint arXiv:2301.07945*, 2023.

[44] P. Jiang, F. Liu, and Y. Song, "A hybrid forecasting model based on date-framework strategy and improved feature selection technology for short-term load forecasting," *Energy*, vol. 119, pp. 694–709, 2017.

[45] R. Jiang, Z. Wang, J. Yong, P. Jeph, Q. Chen, Y. Kobayashi, X. Song, S. Fukushima, and T. Suzumura, "Spatio-temporal meta-graph learning for traffic forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 7, 2023, pp. 8078–8086.

[46] D. Jin, J. Shi, R. Wang, Y. Li, Y. Huang, and Y.-B. Yang, "Trafformer: unify time and space in traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, 2023, pp. 8114–8122.

[47] X. Jin, Y. Park, D. C. Maddix, Y. Wang, and X. Yan, "Attention-based domain adaptation for time series forecasting," *arXiv preprint arXiv:2102.06828*, 2021.

[48] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, "Hard negative mixing for contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 798–21 809, 2020.

[49] N. Karani, E. Erdil, K. Chaitanya, and E. Konukoglu, "Test-time adaptable neural networks for robust medical image segmentation," *Medical Image Analysis*, vol. 68, p. 101907, 2021.

[50] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020.

[51] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

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

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[54] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "Fnet: Mixing tokens with fourier transforms," *arXiv preprint arXiv:2105.03824*, 2021.

[55] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image

pre-training with frozen image encoders and large language models," *arXiv preprint arXiv:2301.12597*, 2023.

[56] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4189–4196.

[57] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in Neural Information Processing Systems*, vol. 32, pp. 5243–5253, 2019.

[58] S. Li, J. Zhou, J. Liu, T. Xu, E. Chen, and H. Xiong, "Multi-temporal relationship inference in urban areas," *arXiv preprint arXiv:2306.08921*, 2023.

[59] X. Li, G. Pan, Z. Wu, G. Qi, S. Li, D. Zhang, W. Zhang, and Z. Wang, "Prediction of urban human mobility using large-scale taxi traces and its applications," *Frontiers of Computer Science*, vol. 6, no. 1, pp. 111–121, 2012.

[60] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[61] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *International conference on machine learning*. PMLR, 2020, pp. 6028–6039.

[62] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "Geoman: Multi-level attention networks for geo-sensory time series prediction." in *IJCAI*, vol. 2018, 2018, pp. 3428–3434.

[63] B. Liao, J. Zhang, C. Wu, D. McIlwraith, T. Chen, S. Yang, Y. Guo, and F. Wu, "Deep sequence learning with auxiliary information for traffic prediction," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 537–546.

[64] Y. Liu, P. Kothari, B. Van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, "Ttt++: When does self-supervised test-time training fail or thrive?" *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 808–21 820, 2021.

[65] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105.

[66] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[67] S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, and A. Zeng, "Large language models as general pattern machines," *arXiv preprint arXiv:2307.04721*, 2023.

[68] S. Moosavi, M. H. Samavatian, S. Parthasarathy, R. Teodorescu, and R. Ramnath, "Accident risk prediction based on heterogeneous sparse data: New dataset and insights," in *Proceedings of the 27th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2019, pp. 33–42.

[69] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi–passenger demand using streaming data," *IEEE*

Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1393–1402, 2013.

[70] S. L. Mullally and E. A. Maguire, "Memory, imagination, and predicting the future: a common brain mechanism?" *The Neuroscientist*, vol. 20, no. 3, pp. 220–234, 2014.

[71] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[72] C. Pal, S. Hirayama, S. Narahari, M. Jeyabharath, G. Prakash, and V. Kulothungan, "An insight of world health organization (who) accident database by cluster analysis with self-organizing map (som)," *Traffic injury prevention*, vol. 19, no. sup1, pp. S15–S20, 2018.

[73] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "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*, 2019, pp. 1720–1730.

[74] M. Poli, S. Massaroli, F. Berto, J. Park, T. Dao, C. Ré, and S. Ermon, "Transform once: Efficient operator learning in frequency domain," *arXiv preprint arXiv:2211.14453*, 2022.

[75] S. Qin, F. Liu, C. Wang, Y. Song, and J. Qu, "Spatial-temporal analysis and projection of extreme particulate matter (pm10 and pm2. 5) levels using association rules: A case study of the jing-jin-ji region, china," *Atmospheric Environment*, vol. 120, pp. 339–350, 2015.

[76] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[77] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[78] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5301–5310.

[79] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances in neural information processing systems*, vol. 20, 2007.

[80] Y. Rao, W. Zhao, Z. Zhu, J. Lu, and J. Zhou, "Global filter networks for image classification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 980–993, 2021.

[81] D. L. Schacter, D. R. Addis, D. Hassabis, V. C. Martin, R. N. Spreng, and K. K. Szpunar, "The future of memory: remembering, imagining, and the brain," *Neuron*, vol. 76, no. 4, pp. 677–694, 2012.

[82] C. Shang, J. Chen, and J. Bi, "Discrete graph structure learning for forecasting multiple time series," *arXiv preprint arXiv:2101.06861*, 2021.

[83] S. Shekhar and B. M. Williams, "Adaptive seasonal time series models for forecasting short-term traffic flow," *Transportation Research Record*, vol. 2024, no. 1, pp. 116–125, 2007.

[84] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[85] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl *et al.*, "Large language models encode clinical knowledge," *arXiv preprint arXiv:2212.13138*, 2022.

[86] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 914–921.

[87] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *International conference on machine learning*. PMLR, 2020, pp. 9229–9248.

[88] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[89] Y. Tang, Z. Jiang, Z. Xie, Y. Cao, Z. Zhang, P. H. Torr, and H. Hu, "Breaking shortcut: Exploring fully convolutional cycle-consistency for video correspondence learning," *arXiv preprint arXiv:2105.05838*, 2021.

[90] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

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

[92] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating

hidden states," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6438–6447.

[93] V. Verma, T. Luong, K. Kawaguchi, H. Pham, and Q. Le, "Towards domain-agnostic contrastive learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 530–10 541.

[94] C. Villani *et al.*, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[95] B. Wang, Y. Lin, S. Guo, and H. Wan, "Gsnet: Learning spatial-temporal correlations from geographical and semantic aspects for traffic accident risk forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4402–4409.

[96] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," *arXiv preprint arXiv:2006.10726*, 2020.

[97] S. Wang, J. Zhang, J. Li, H. Miao, and J. Cao, "Traffic accident risk prediction via multi-view multi-task spatio-temporal networks," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[98] S. Wang, M. Zhang, H. Miao, Z. Peng, and P. S. Yu, "Multivariate correlation-aware spatio-temporal graph convolutional networks for multi-scale traffic prediction," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 3, pp. 1–22, 2022.

[99] X. Wang, A. Jabri, and A. A. Efros, "Learning correspondence from the cycle-consistency of time," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2566–2576.

[100] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu,

"Traffic flow prediction via spatial temporal graph neural network," in *Proceedings of The Web Conference 2020*, 2020, pp. 1082–1092.

[101] W. W. Wei, "Time series analysis," in *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*, 2006.

[102] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *Journal of transportation engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[103] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, "Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting," *arXiv preprint arXiv:2202.01575*, 2022.

[104] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 419–22 430, 2021.

[105] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "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*, 2020, pp. 753–763.

[106] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *arXiv preprint arXiv:1906.00121*, 2019.

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

[108] Y. Xu, Y. Han, R. Hong, and Q. Tian, "Sequential video vlad: Training the aggregation locally and temporally," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 4933–4944, 2018.

[109] H. Xue, B. P. Voutharoja, and F. D. Salim, "Leveraging language foundation models for human mobility forecasting," in *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, 2022, pp. 1–9.

[110] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[111] J. Ye, J. Zhao, K. Ye, and C. Xu, "How to build a graph-based deep learning architecture in traffic domain: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[112] J. Ye, Z. Liu, B. Du, L. Sun, W. Li, Y. Fu, and H. Xiong, "Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2296–2306.

[113] J. Ye, L. Sun, B. Du, Y. Fu, X. Tong, and H. Xiong, "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*, 2019, pp. 305–313.

[114] J. Ye, L. Sun, B. Du, Y. Fu, and H. Xiong, "Coupled layer-wise graph convolution for transportation demand prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4617–4625.

[115] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proceedings of the 27th*

International Joint Conference on Artificial Intelligence, 2018, pp. 3634–3640.

[116] H. Yuan and G. Li, "A survey of traffic prediction: from spatio-temporal data to intelligent transportation," *Data Science and Engineering*, vol. 6, no. 1, pp. 63–85, 2021.

[117] H. Yuan, G. Li, Z. Bao, and L. Feng, "An effective joint prediction model for travel demands and traffic flows," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 348–359.

[118] Z. Yuan, X. Zhou, and T. Yang, "Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 984–992.

[119] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2018, pp. 231–240.

[120] D. Zhang, L. Yao, K. Chen, S. Wang, X. Chang, and Y. Liu, "Making sense of spatio-temporal preserving representations for eeg-based human intention recognition," *IEEE transactions on cybernetics*, vol. 50, no. 7, pp. 3033–3044, 2019.

[121] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[122] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proceedings of the AAAI conference on*

artificial intelligence, vol. 31, no. 1, 2017.

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

[124] L. Zhang, C. Aggarwal, and G.-J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 2141–2149.

[125] X. Zhang, Z. Zhao, T. Tsiligkaridis, and M. Zitnik, "Self-supervised contrastive pre-training for time series via time-frequency consistency," *arXiv preprint arXiv:2206.08496*, 2022.

[126] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *International Conference on Learning Representations*, 2023.

[127] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1234–1241.

[128] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

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

[130] Z. Zhou, "Attention based stack resnet for citywide traffic accident prediction," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2019, pp. 369–370.

[131] Z. Zhou, Y. Wang, X. Xie, L. Chen, and H. Liu, "Riskoracle: a minute-level citywide traffic accident forecasting framework," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1258–1265.

[132] L. Zhu, Z. Xu, and Y. Yang, "Bidirectional multirate reconstruction for temporal modeling in videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2653–2662.

[133] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," *Modeling Financial Time Series with S-Plus®*, pp. 385–429, 2006.