Enhancing Dependency Dynamics in Traffic Flow Forecasting via Graph Risk Bootstrap

Gao, Qiang; Wang, Zizheng; Huang, Li; Trajcevski, Goce; Zhang, Kunpeng; Chen, Xueqin

DOI
10.1145/3678717.3691237

Publication date
2024

Document Version
Final published version

Published in
SIGSPATIAL '24

Citation (APA)
Gao, Q., Wang, Z., Huang, L., Trajcevski, G., Zhang, K., & Chen, X. (2024). Enhancing Dependency Dynamics in Traffic Flow Forecasting via Graph Risk Bootstrap. In M. A. Nascimento, L. Xiong, A. Zufle, Y.-Y. Chiang, A. Eldawy, & P. Kröger (Eds.), *SIGSPATIAL '24: Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems* (pp. 147-159). ACM.
https://doi.org/10.1145/3678717.3691237

Enhancing Dependency Dynamics in Traffic Flow Forecasting via Graph Risk Bootstrap

Qiang Gao
Southwestern University of Finance and Economics
Chengdu, China
qianggao@swufe.edu.cn

Zizheng Wang
Southwestern University of Finance and Economics
Chengdu, China
223081200032@smail.swufe.edu.cn

Li Huang*
Southwestern University of Finance and Economics
Chengdu, China
lihuang@swufe.edu.cn

Goce Trajcevski
Iowa State University
Ames, Iowa, USA
gocet25@iastate.edu

Kunpeng Zhang
University of Maryland
College Park, USA
kpzhang@umd.edu

Xueqin Chen
Delft University of Technology
Delft, The Netherlands
nedchen0728@gmail.com

Abstract

Graph neural networks, as well as attention mechanisms, have gained widespread popularity for traffic flow forecasting due to their capacity to incorporate the complicated interactions behind flow dynamics. However, existing solutions either formulate a graph-based skeleton with narrow (e.g., static) interaction capture or build the spatiotemporal (e.g., dynamic) attention without proper comprehension of diverse risks, which inevitably burdens the generalization of high-accuracy traffic trends. In this study, we introduce **Gboot** (**G**raph **boot**strap) enhancement framework for traffic flow forecasting. Gboot takes the traffic flow forecasting problem from a dependency dynamic learning perspective by treating each traffic sensor as the graph node while regarding the observed flows at each sensor as the node feature. In addition to exposing the explicit spatial connectivity behind traffic flows, we hierarchically devise temporal-aware and factual-aware graph learning blocks to consider temporal interactive dynamics and factual interactive dynamics. The former shows the trend dependencies behind flow signals and the latter uncovers different views of traffic situations (e.g., current observation vs. historical observation). More importantly, we present a Dual-view Bootstrap (DvBoot) mechanism in Gboot, which includes both risk-free and risk-aware stands. DvBoot attempts to flexibly align these two views in the latent space to enhance the generalization capability of capturing dynamic dependencies. Experiments on several real-world traffic datasets demonstrate the superiority of our Gboot over representative approaches.

CCS Concepts

• **Information systems** → **Geographic information systems**; **Sensor networks**.

*Corresponding Author (lihuang@swufe.edu.cn)

SIGSPATIAL '24, October 29-November 1, 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-1107-7/24/10
https://doi.org/10.1145/3678717.3691237

Keywords

traffic flow forecasting, dependency dynamics, bootstrap learning, exponential moving average, risk enhancement

ACM Reference Format:
Qiang Gao, Zizheng Wang, Li Huang, Goce Trajcevski, Kunpeng Zhang, and Xueqin Chen. 2024. Enhancing Dependency Dynamics in Traffic Flow Forecasting via Graph Risk Bootstrap. In *The 32nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24), October 29-November 1, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3678717.3691237

1 Introduction

Traffic management is one of the critical aspects of intelligent transportation systems (ITS) [19, 21, 40]. The ubiquity of sensor networks in urban areas provides an unprecedented opportunity to gather traffic-related data in specific locations at any time, offering insight into historical traffic regularities that enable prediction of future tendencies. Traffic flow forecasting also plays a significant role in urban management, as it can help improve regulatory capacities, risk assessment, improved trip experience, etc. [18, 19, 32, 45].

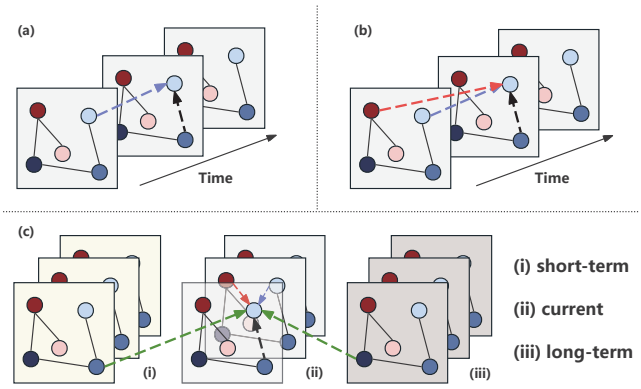


Figure 1: Recent solutions ((a) and (b)) and ours (c).

Typically, raw traffic data can be modeled as spatial and/or temporal graphs, where the actual *sensors* along the road network correspond to *nodes*; and *dependencies* (or *proximities*) create *connectivity (i.e., edges)* between different nodes (cf. Fig. 1). Based on

this, graph neural network (GNN)-based solutions have been proposed in the field of traffic forecasting [2, 10, 17, 22, 46]. The dominant paradigm in traffic flow data modeling using GNNs (usually called spatial-temporal GNNs) involves the building of a spatial-aware graph to delineate the nature of connectivity (i.e., geospatial correlations) within the road network *and* explore the temporal dependencies through widely used Convolutional Neural Networks (CNNs) [8, 12], Recurrent Neural Networks (RNNs) [2, 24] or self-attention networks [45, 46]. For example, Fig. 1(a) illustrates the potential correlations of traffic conditions over time, where the purple dashed line denotes the temporal correlation while the black dashed line indicates the spatial correlation. In this context, the ST-GDN framework builds a spatial-aware graph to expose the connective relations of spatial regions and employs the self-attention network to encode the temporal dynamics presented in [45]. STGODE [8] extends multi-layer Graph Convolutional Networks (GCNs) with tensor-based ordinary differential equations, owning the capacity to capture long-range spatial correlations. In a nutshell, they initially investigate the impact of spatially adjacent sensors or nodes within the same time step and, subsequently, focus on capturing the temporal correlations associated with each node across different time steps.

In addition to separately modeling the spatial and temporal dependencies, another line of recent works attempts to explore more complex spatial-temporal interactions (e.g., the red dashed line in Fig. 1(b)) due to the existence of synchronous spatial-temporal correlations [30]. That is, each node could share a positive signal with its spatially adjacent nodes at the next time step(s), i.e., localized spatial-temporal correlations. To this end, [30] introduced Spatial-Temporal Synchronous Graph Convolutional Networks (STSGCN) to synchronously capture the localized spatial-temporal correlations directly and [22] introduced a CNN-based spatial-temporal fusion GNN, called STFGNN, to extract confidential spatial-temporal dependencies. Gated convolution on graph-aware dynamics was devised to expose the long-range spatial-temporal correlations – DSTAGNN [21] structures a spatial-temporal attention module to explore the spatial-temporal interactions in a road network by extending the GCNs. The above solutions demonstrated that consideration of the dynamics of spatial-temporal interactions enhances the ability to predict the future trends of different traffic signals.

Despite the significant breakthroughs achieved in various GNN-based efforts for traffic flow forecasting, we argue that present solutions grapple with two predominant challenges.

C1: In addition to spatial-temporal interaction, the current observed flow volume for each sensor is not only influenced by the local (e.g., adjacent time step) or global (e.g., non-Euclidean distance) spatial-temporal structure but also depends on historical observations. For example, a sensor at the adjacent instants in the past will generally have comparable or interactive patterns, exhibiting different prior views of traffic states. This inspiration is drawn from the fact that historical traffic records exhibit a high degree of continuity owing to factors like urban commuting and daily life patterns. Thus, sensor observations at any given instant are potentially significant in relation to their historical counterparts. We conjecture that exposing the factual interaction between different flow situations (current observation vs. historical observation) on the basis of spatial connectivity could boost the perception of future trends. In addition, traffic flows are typically multivariate time-series data. While existing solutions pay attention to capturing adjacent dependencies (e.g., using RNNs) or global dependencies (i.e., using attentions) - they rarely consider different ranges of dependency learning, yielding a narrow receptive field for handling flow dynamics.

C2: Due to urban congestion, commuting preferences, sensor fluctuations, and other uncertain irregularities, current solutions heavily depend on stable (i.e., risk-free) pattern data while lacking attention to complex traffic risks. This inevitably burdens the generalization of high-accuracy traffic trends and raises the representation gaps between accuracy and robustness. Recent advances in data augmentation techniques such as contrasting augmentation (i.e., producing multiple uncertain views to the actual instance) [15, 23, 44] can alleviate the mentioned representation gap problem (to an extent). However, they rely on the tremendous negative pairs by comparing each instance (e.g., traffic flow observation) with many other examples (e.g., augmented observations) to work well in the latent (representation) space [5]. For instance, recent contrastive learning solutions handcraft hundreds or even thousands of negative pairs and then employ Noise Contrastive Estimation (NCE) to make similar instances closer in the representation space while dissimilar instances are (relatively) further away [23]. But this could confront either collapsed representations or unstable representations due to the uncertain augmentations on original data. Recent experimental study [41] demonstrated that these augmentations with negative pairs in representation learning are trivial. In sum, the challenge of effectively using risk-aware views to enhance model generalization capabilities still persists.

To address the above challenges, we introduce **Gboot** – a novel **G**raph **boot**strap enhancement framework rooted in a dependency dynamic learning paradigm. It first builds a traffic graph derived from the road network, treats each traffic sensor as the graph node and the observed flows at each sensor as the node feature. To expose multiple dependency dynamics, we devise a *Traffic Dependency Learner* (TDL) to consider two interactive dependencies over the spatial connectivity behind the traffic flow records. Specifically, we hierarchically manage temporal and factual-aware graph learning blocks in TDL to capture temporal and factual interactive dynamics. In particular, we combine multi-scaled historical observations of each sensor with the current one as the prior views of traffic situations, primarily seeking to fully augment the factual interaction between different flow situations (cf. Fig. 1(c), where each part describes different graph-aware flow situations and green dashed lines indicate the factual interactions). To enhance risk-aware compatibility, we introduce a *Dual-view Bootstrap* (DvBoot) mechanism in Gboot, which includes both *risk-free* and *risk-aware* stands. DvBoot attempts to flexibly align these two views in the latent space to enhance the generalization ability of flow dynamics learning. Finally, our main contributions can be summarized as follows:

- Our TDL in Gboot primarily contains two blocks, i.e., Temporal-aware Graph Learning (TGL) and Factual-aware Graph Learning (FGL). Specifically, TGL attempts to capture temporal interactive dynamics over spatial connectivity learning, while FGL aims to expose factual interactive dynamics over spatial connectivity learning, i.e., current observation vs. (short/long-term) historical observation. More importantly, in each block, we involve latent

capture with a multi-scaled gated convolution network to handle different ranges of temporal dynamics or factual dynamics.
- Motivated by recent exponential moving average (EMA) theory, our DvBoot treats the TDL handling the raw flow data as a deterministic (risk-free) view while using the TDL with different parameter settings to handle the risk-aware view, motivating a variety of risk conditions, including missing facts, missing spatial connectivity, and flow noise. The dual-view design does not need to involve tremendous negative instances, enhancing the model's robustness by directly immigrating the predictive representation gaps between risk-free and risk-aware views.
- Experiments on several real-world traffic datasets demonstrate the superiority of our Gboot over representative approaches.

2 Related Work

We now overview recent solutions for addressing the traffic flow forecasting problem–and we position our results in that context.

Conventional methods for traffic flow forecasting mainly focus on employing the auto-regressive integrated moving average model (ARIMA) [20] and its variants, which cannot handle spatial-temporal dependencies such as sensors' location and seasonal factors. During the past decade, by virtue of deep learning, researchers have developed various methods to combat this limitation. Initially, many approaches utilize Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) to model temporal correlations [28] and spatial dependencies [43], respectively.

Despite their promising results, they still faced limitations in fully exploring the complex spatial patterns inherent in unstructured road networks. Recently, motivated by the successful application of Graph Neural Networks (GNNs) in addressing non-Euclidean spatial correlations, researchers have begun to incorporate GNNs into their model development [18]. DCRNN [24] as the first GNN-based model for traffic flow prediction, which melds GRU with graph diffusion convolution networks to simultaneously capture spatial-temporal dependencies. STGCN [40] uses spatial-graph convolution and temporal-gated convolution to capture the spatial and temporal dependencies, respectively. Following this, to enhance model performance, certain studies have either integrated attention mechanisms [34] from the fields of Computer Vision (CV) and Natural Language Processing (NLP) into spatiotemporal graph modeling [10, 12, 27, 35, 47] or constructed diverse spatial graphs via considering various types of connections, such as semantic connection [1, 8], and edge interaction patterns [6]. Moreover, some studies focus on integrating new paradigms into GNNs to improve model performance. For instance, STGODE [8] incorporated Ordinary Differential Equations (ODE) into GCNs, based on the combination of both semantic and road graphs. And ST-SSL [15] employs self-supervised learning paradigms to enhance traffic pattern representations, ensuring they reflect the inherent spatial and temporal heterogeneity. However, the spatial dependencies captured by these models do not adequately reflect the inherent dynamics. To address this, DSTAGNN [21] designs a spatial-temporal attention module and a multi-receptive field-gated convolution to effectively learn the dynamic associations between nodes within the road network.

What separates our work from the existing literature is that: (1) We take the traffic flow forecasting problem from a dependency dynamic learning perspective by treating the observed flows at each sensor as the node feature while exposing both temporal and factual dynamics over the basis of spatial connectivity; (2) We build a dual-view (risk-free vs. risk-aware) bootstrap mechanism to handle the risks that could be raised in the flow data, primarily seeking to enhance the generalization ability of flow dynamic learning.

3 Methodology

We now provide basic definitions (e.g., road network and traffic flow observations) and a high-level overview of the Gboot structure, followed by a detailed discussion of the respective main modules.

3.1 Preliminaries and Architecture Overview

Definition 1 (Road Network). *Let traffic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a road network in a city/region, where $\mathcal{V}$ represents a set of $n = |\mathcal{V}|$ nodes (e.g., observation stations/sensors) within the road network, and $\mathcal{E}$ is a set of edges that display the spatial connectivity between different observation sensors. Besides, let $\mathcal{A} \in [0,1]^{n\times n}$ refer to the adjacency matrix exposing spatial connectivity between nodes.*

Definition 2 (Traffic Flow Observations). *Let three-way tensor $\boldsymbol{X}^{t-\omega+1:t} \in \mathbb{R}^{n\times c\times \omega}$ represent traffic flow data with the observation window ω, where any $\boldsymbol{X}_\tau \in \mathbb{R}^{n\times c}$ is a graph signal depicting the traffic observations of n sensors with c situations at time step τ. Herein, c refers to the types of traffic situations/conditions, e.g., current traffic volume, short/long-term historical average traffic volume, etc.*

Since traffic flow is characterized by typical periodicity, an excessive focus on long-history dynamics (denoted as extra *situations*) can cause modeling complexity problems in addition to additional unstable training risks. Hence, in addition to the current traffic volume (that is, $c = 1$), this study considers the average volumes of the preceding half hour and the preceding hour relative to the current flow as short- and long-term situations. For instance, we assume that $[x_{t-\omega+1}, x_{t-\omega+2}, \cdots, x_t]$ refers to the current situation (traffic volume) of a sensor. The short-term situation can be defined as:

$$\left[\alpha[\sum_{i=0}^{\frac{\omega}{2}-1}(x_{t-\omega+1-i})], \alpha[\sum_{i=0}^{\frac{\omega}{2}-1}(x_{t-\omega+2-i})], \cdots, \alpha[\sum_{i=0}^{\frac{\omega}{2}-1}(x_{t-i})]\right]. \tag{1}$$

Herein, α refers to the average function. In this way, we can also obtain a long-term situation (the preceding hour) of this sensor. In the end, the core problem addressed in this work can be defined as:

Definition 3 (Traffic Flow Forecasting). *Given historical traffic flow records $\boldsymbol{X}^{t-\omega+1:t} \in \mathbb{R}^{n\times c\times \omega}$, we aim to learn a forecasting model $\mathcal{M}$ with parameters Θ to predict the traffic observations (i.e., traffic volumes) of next ω time steps $\widehat{\boldsymbol{Y}}^{t+1:t+\omega} \in \mathbb{R}^{n\times \omega}$ on graph $\mathcal{G}$:*

$$\widehat{\boldsymbol{Y}}^{t+1:t+\omega} = \mathcal{M}_\Theta(\boldsymbol{X}^{t-\omega+1:t}; \mathcal{G}). \tag{2}$$

Note that for simplicity, the superscript of $\boldsymbol{X}^{t-\omega+1:t}$ will be omitted in the following sections. Fig. 2 illustrates the network skeleton of Gboot, which contains two modules: Traffic Dependency Learner (TDL) and Traffic Flow Predictor (TFP). TDL comprises two blocks where Temporal-aware Graph Learning (TGL) attempts to capture temporal interactive dynamics over spatial connectivity learning while Factual-aware Graph Learning (FGL) explores factual interactions in the latent space. In Fig. 2, details of Temporal/Factual

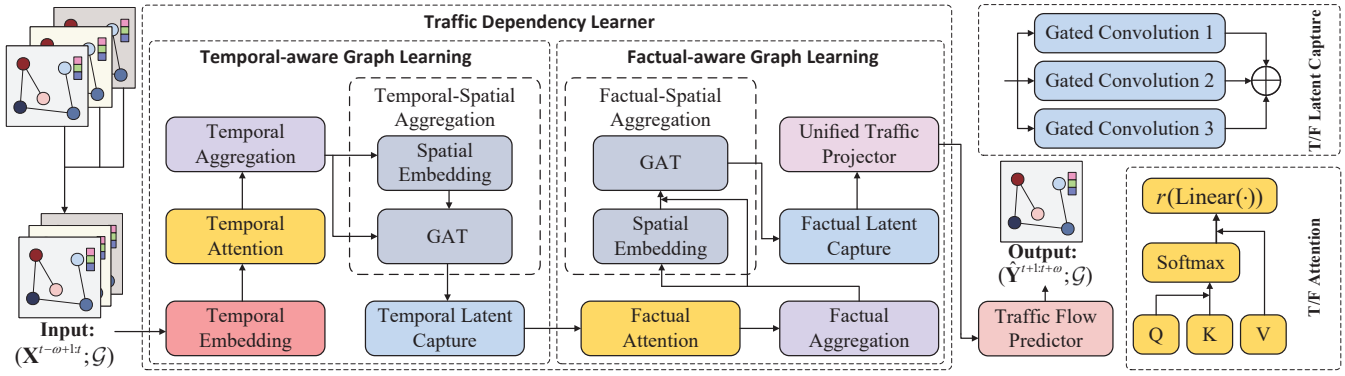


Figure 2: The network skeleton of our proposed Gboot.

(or T/F) Latent Capture and Attention are shown in the upper and lower right corners, respectively. Besides, we will employ our Dual-view Bootstrap (DvBoot) mechanism to produce the Siamese of the TDL, which can be treated as a risk-aware view (detailed next).

3.2 Traffic Dependency Learner (TDL)

We aim to explore two interactive dependencies over the spatial connectivity behind the traffic flow records. Specifically, TDL contains two blocks, including Temporal-aware Graph Learning (TGL) to capture temporal interactive dynamics, Factual-aware Graph Learning (FGL) to explore factual interactive dynamics.

We note that in our context, when it comes to *traffic flow embedding*, we treat the historical flow observations of a node at different time steps as the node feature which would, in a sense, blur the time attribute of each node. For that, we employ a positional embedding [30] to specify the different time step signals, aiming at enhancing the ability to model the temporal correlations. We select the simple embedding method with absolute positions [9] to tackle the input $\boldsymbol{X} \in \mathbb{R}^{n\times c\times\omega}$, which can be represented as follows:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{X} + \text{broad}(\boldsymbol{X}, \boldsymbol{T}), \tag{3}$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times c\times\omega}$, $\boldsymbol{T} \in \mathbb{R}^{c\times\omega}$ is a trainable temporal embedding with random initialization, and broad() is the *Broadcast* operation.

3.2.1 Temporal-aware Graph Learning (TGL). TGL is responsible for capturing temporal interactive dynamics on the basis of spatial connectivity. Specifically, it is stacked with multiple layers to conduct different roles of knowledge distillation, as detailed below.

Temporal Attention: We employ the popular self-attention mechanism [34] with temporal view to tackle the interactive contributions behind traffic flow representations $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n\times c\times\omega}$. In detail, we first shift the view to the temporal aspect, i.e., $\boldsymbol{\mathcal{X}}^t = r(\boldsymbol{\mathcal{X}})$, where $\boldsymbol{\mathcal{X}}^t \in \mathbb{R}^{c\times\omega\times n}$ and $r()$ refers to the *Reshape* operation. Next, we operate a standard multi-head self-attention layer to produce the latent states. For a head $h \in [1, 2, \cdots, H]$, it can be denoted as:

$$\boldsymbol{O}^{t,h} = \text{Softmax}(\frac{\boldsymbol{Q}^{t,h}(\boldsymbol{K}^{t,h})^\top}{\sqrt{d_t}})\boldsymbol{V}^{t,h}, \tag{4}$$

where $\boldsymbol{Q}^{t,h} = \boldsymbol{\mathcal{X}}^t\boldsymbol{W}_Q^{t,h}$, $\boldsymbol{K}^{t,h} = \boldsymbol{\mathcal{X}}^t\boldsymbol{W}_K^{t,h}$, and $\boldsymbol{V}^{t,h} = \boldsymbol{\mathcal{X}}^t\boldsymbol{W}_V^{t,h}$ refer to the query, key, and value in h-th head of the self-attention network, respectively. Any $\boldsymbol{W}_*^{t,h} \in \mathbb{R}^{n\times d_t}$ is a trainable matrix. Then, we concatenate these head outputs into a unified representation:

$$\boldsymbol{O}^t = [\boldsymbol{O}^{t,1}; \boldsymbol{O}^{t,2}; \cdots, \boldsymbol{O}^{t,H}]. \tag{5}$$

Now we shift the results of $\boldsymbol{O}^t$ to the original view and align it to $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n\times c\times\omega}$ with a simple fully connected network (denoted as 'Linear'). To this end, we employ a widely used residual operation and layer normalization (denoted as 'LayerNorm') to obtain the final output $\boldsymbol{O}^t \in \mathbb{R}^{n\times c\times\omega}$, which can be summarized as follows:

$$\boldsymbol{O}^t = \text{LayerNorm}(r(\text{Linear}(\boldsymbol{O}^t)) + \boldsymbol{\mathcal{X}}). \tag{6}$$

Temporal Aggregation: For observed n nodes in the road network, we now turn to aggregate the temporal semantics into these nodes. Specifically, we operate a convolutional layer with filter $\boldsymbol{I}_t$ (the kernel size is $1 \times c$, the input channel is ω, and the output channel is d_S) to tackle the output $\boldsymbol{O}^t \in \mathbb{R}^{n\times c\times\omega}$, summarized as follows:

$$\boldsymbol{S}^t = \boldsymbol{O}^t * \boldsymbol{I}^t, \tag{7}$$

where $\boldsymbol{S}^t \in \mathbb{R}^{n\times d_S}$ and $*$ represents the convolution operation.

Temporal-Spatial Aggregation: To capture the topological (i.e., spatial) structure with temporal perception, we naturally select GNNs to aggregate the node information. We also equip the positional embedding operation to the input $\boldsymbol{S}^t$ to specify the spatial attribute on graph $\mathcal{G}$:

$$\boldsymbol{\mathcal{S}}^t = \boldsymbol{S}^t + broad(\boldsymbol{S}^t, \Gamma^t), \tag{8}$$

where $\Gamma^t \in \mathbb{R}^{d_S}$ is a learnable spatial embedding regarding $\boldsymbol{S}^t$. Next, we employ the graph attention network (GAT) [3] to specify the different contributions between different nodes. For instance, given a node $i \in \mathcal{V}$ and a node $j \in \mathcal{V}$, GAT attempts to explore the attentive correlation between their node representations, i.e., $s_i^t \in \boldsymbol{\mathcal{S}}^t$ and $s_j^t \in \boldsymbol{\mathcal{S}}^t$, which can be summarized as follows:

$$e(s_i^t, s_j^t) = a^{t\top}\text{LeakyReLU}\left(\boldsymbol{W}_t^e \cdot \left[s_i^t \,\|\, s_j^t\right]\right), \tag{9}$$

$$\beta_{ij}^t = \text{Softmax}(e(s_i^t, s_j^t)) = \frac{\exp(e(s_i^t, s_j^t))}{\sum_{j'\in\mathcal{A}_i}\exp(e(s_i^t, s_{j'}^t))}, \tag{10}$$

$$s_i^{ts} = \text{LeakyReLU}(\sum_{j\in\mathcal{A}_i}\beta_{ij}^t \cdot \boldsymbol{W}_t^s s_j^t), \tag{11}$$

where a^t is a learnable vector with random initialization. $\boldsymbol{W}_t^e \in \mathbb{R}^{\omega\times 2d_S}$ and $\boldsymbol{W}_t^s \in \mathbb{R}^{\omega\times d_S}$ are trainable matrices. $\mathcal{A}_i$ indicates the actual spatial neighbors of node i according to the adjacent matrix

$\mathcal{A}$. In practice, we use multi-head GAT to generalize the learning capability of the model and align the output with the original input traffic flow tensor scale for the purpose of conducting the following residual operation, i.e., the head number is c. As such, the final output of this layer can be denoted as $\boldsymbol{\mathcal{S}}^{ts} \in \mathbb{R}^{n\times c\times\omega}$.

Temporal Latent Capture: Motivated by [7, 21], we operate the multi-scaled gated convolution network to tackle the output of the above layer, primarily seeking to expose the temporal dynamics in the latent space with different ranges. Specifically, it contains three gated convolution operators with different scales (i.e., receptive fields/filters), uncovering different ranges of temporal dynamics. For instance, each gated convolution operator can be denoted as:

$$\boldsymbol{\mathcal{Z}}^{t} = \psi(r(\boldsymbol{\mathcal{S}}^{ts}) * \boldsymbol{U}^{t}) \otimes \sigma(r(\boldsymbol{\mathcal{S}}^{ts}) * \boldsymbol{U}_{g}^{t}), \tag{12}$$

where $\otimes$ is the element-wise product, ψ is an activation function (herein it is a *Tanh* function for the purpose of retaining the non-linearity), $r(\boldsymbol{\mathcal{S}}^{ts}) \in \mathbb{R}^{c\times n\times\omega}$, and σ is the gated function, i.e., *Sigmoid*. $\boldsymbol{U}^{t}$ with kernel size $1\times z$ and $\boldsymbol{U}_{g}^{t}$ with kernel size $1\times z$ are convolution filters. Likewise, we concatenate the outputs of different scaled convolutions with residual operation, denoted as:

$$\boldsymbol{Z}_{out}^{t} = [\boldsymbol{Z}^{t,1}; \boldsymbol{Z}^{t,2}; \boldsymbol{Z}^{t,3}], \tag{13}$$

$$\boldsymbol{\mathcal{Z}}^{t} = \text{LayerNorm}(r(Linear(\boldsymbol{Z}_{out}^{t})) + \boldsymbol{\mathcal{S}}^{ts}). \tag{14}$$

We will employ $\boldsymbol{\mathcal{Z}}^{t} \in \mathbb{R}^{n\times c\times\omega}$ as the input of the following FGL.

3.2.2 Factual-aware Graph Learning (FGL). Our devised FGL attempts to explore the situational dependencies behind the flow dynamics in the latent space, uncovering the factual interaction between different flow situations (flow views) over the basis of spatial connectivity. Similar to TGL, it also contains multiple layers for handling different types of interactive knowledge.

Factual Attention: Similar to the *Temporal Attention* mechanism in the TGL block, we also employ a multi-head attention network to explore the interactive contributions between different facts. Hence, we first shift the sight to the factual view, i.e., $\boldsymbol{\mathcal{Z}}^{f} = r(\boldsymbol{\mathcal{Z}}^{t})$, where $\boldsymbol{\mathcal{Z}}^{f} \in \mathbb{R}^{\omega\times c\times n}$. Subsequently, we operate a standard multi-head self-attention neural network to produce the latent representations. It is worth noting that we do not engage in positional embedding here, as we treat these latent facts equally as different views of the traffic situation, which contrasts with the temporal semantic refinement. For a head $h \in [1, 2, \cdots, H]$, it can be denoted as:

$$\boldsymbol{O}^{f,h} = \text{Softmax}(\frac{\boldsymbol{Q}^{f,h}(\boldsymbol{K}^{f,h})^{\top}}{\sqrt{d_f}})\boldsymbol{V}^{f,h}, \tag{15}$$

where $\boldsymbol{Q}^{f,h} = \boldsymbol{\mathcal{Z}}^{f}\boldsymbol{W}_{Q}^{f,h}$, $\boldsymbol{K}^{f,h} = \boldsymbol{\mathcal{Z}}^{f}\boldsymbol{W}_{K}^{f,h}$, and $\boldsymbol{V}^{f,h} = \boldsymbol{\mathcal{Z}}^{f}\boldsymbol{W}_{V}^{f,h}$ refer to the query, key, and value in the h-head of the self-attention layer, respectively. Any $\boldsymbol{W}_{*}^{f,h} \in \mathbb{R}^{n\times d_f}$ is a trainable matrix. We also concatenate these head outputs into a unified representation:

$$\boldsymbol{O}^{f} = [\boldsymbol{O}^{f,1}; \boldsymbol{O}^{f,2}; \cdots; \boldsymbol{O}^{f,H}]. \tag{16}$$

Then, we align it to $\boldsymbol{\mathcal{Z}}^{t}$ with a fully connected network. And we also employ a residual operator and layer normalization to obtain the final output $\boldsymbol{O}^{f} \in \mathbb{R}^{n\times c\times\omega}$, summarized as follows:

$$\boldsymbol{O}^{f} = \text{LayerNorm}(r(\text{Linear}(\boldsymbol{O}^{f})) + \boldsymbol{\mathcal{Z}}^{t}). \tag{17}$$

Factual Aggregation: Herein, we operate another convolutional layer with filter size $1\times c$, input channel ω, and output channel d_P to tackle the above aggregate the factual semantics into different nodes of graph $\mathcal{G}$, which can be summarized as follows:

$$\boldsymbol{P}^{f} = \boldsymbol{O}^{f} * \boldsymbol{I}^{f}, \tag{18}$$

where $\boldsymbol{P}^{f} \in \mathbb{R}^{n\times d_P}$ and $\boldsymbol{I}^{f}$ is the filter in the convolution operator.

Factual-Spatial Aggregation: We select GAT to aggregate the node information with factual perception. First, we employ positional embedding with residual connection to tackle $\boldsymbol{P}^{f}$ as:

$$\boldsymbol{\mathcal{P}}^{f} = \boldsymbol{P}^{f} + broad(\boldsymbol{P}^{f}, \Gamma^{f}), \tag{19}$$

where $\Gamma^{f} \in \mathbb{R}^{d_P}$ is another learnable spatial embedding regarding $\boldsymbol{P}^{f}$. Next, given a node $i \in \mathcal{V}$ and a node $j \in \mathcal{V}$ in graph $\mathcal{G}$, we attempt to explore the attentive correlation between their node representations, i.e., $p_i^{f} \in \boldsymbol{\mathcal{P}}^{f}$ and $p_j^{f} \in \boldsymbol{\mathcal{P}}^{f}$, expressed as follows:

$$e(p_i^{f}, p_j^{f}) = {a^{f}}^{\top}\text{LeakyReLU}\left(\boldsymbol{W}_{f}^{e} \cdot \left[p_i^{f} \parallel p_j^{f}\right]\right), \tag{20}$$

$$\beta_{ij}^{f} = \text{Softmax}(e(p_i^{f}, p_j^{f})) = \frac{\exp(e(p_i^{f}, p_j^{f}))}{\sum_{j'\in\mathcal{A}_i}\exp(e(p_i^{f}, p_{j'}^{f}))}, \tag{21}$$

$$p_i^{fs} = \text{LeakyReLU}(\sum_{j\in\mathcal{A}_i}\beta_{ij}^{f} \cdot \boldsymbol{W}_{f}^{s}p_j^{f}), \tag{22}$$

where a^{f} is a learnable vector with random initialization. $\boldsymbol{W}_{f}^{e} \in \mathbb{R}^{\omega\times 2d_P}$ and $\boldsymbol{W}_{f}^{s} \in \mathbb{R}^{\omega\times d_P}$ are trainable matrices. We also use the multi-head GAT to align the output with the original input scale. As such, the final output of this layer can be denoted as $\boldsymbol{\mathcal{P}}^{fs} \in \mathbb{R}^{n\times c\times\omega}$.

Factual Latent Capture: Similarly, we operate the multi-scaled gated convolution network to expose the factual correlations in the latent space. Specifically, it also contains three gated convolution operators with different receptive fields, uncovering different ranges of factual dynamics. For instance, each gated convolution operator can be denoted as follows:

$$\boldsymbol{\mathcal{Z}}^{f} = \psi(r(\boldsymbol{\mathcal{P}}^{fs}) * \boldsymbol{U}^{f}) \otimes \sigma(r(\boldsymbol{\mathcal{P}}^{fs}) * \boldsymbol{U}_{g}^{f}). \tag{23}$$

Herein, ψ is an empirical SwiGLU activation [29] and $r(\boldsymbol{\mathcal{P}}^{fs}) \in \mathbb{R}^{c\times n\times\omega}$. $\boldsymbol{U}^{f}$ with kernel size $1\times z$ and $\boldsymbol{U}_{g}^{f}$ with kernel size $1\times z$ are convolution filters. We then concatenate the outputs of different scaled convolutions with a residual operation, denoted as:

$$\boldsymbol{Z}_{out}^{f} = [\boldsymbol{Z}^{f,1}; \boldsymbol{Z}^{f,2}; \boldsymbol{Z}^{f,3}], \tag{24}$$

$$\boldsymbol{\mathcal{Z}}^{f} = \text{LayerNorm}(r(Linear(\boldsymbol{Z}_{out}^{f})) + \boldsymbol{\mathcal{P}}^{fs}). \tag{25}$$

Unified Traffic Projector (UTP): It is to formulate the universal representation by combining the outputs of TGL and FGL. Specifically, we first concatenate them by:

$$\boldsymbol{\mathcal{Z}} = [\boldsymbol{\mathcal{Z}}^{f}; \boldsymbol{\mathcal{Z}}^{t}] \in \mathbb{R}^{n\times c\times 2\omega}. \tag{26}$$

Then, we employ a convolutional layer with filter $\boldsymbol{I}$ (the kernel size is $1\times \mathrm{c}$, input channel 2ω, and output channel d_{c}), summarized as:

$$\boldsymbol{C} = \boldsymbol{\mathcal{Z}} * \boldsymbol{I}. \tag{27}$$

The above output $\boldsymbol{C} \in \mathbb{R}^{n\times d_{\mathrm{c}}}$ will be put into the Task Predictor for future flow forecasting.

3.3 Traffic Flow Predictor (TFP)

Given a node $i \in \mathcal{V}$ and a node $j \in \mathcal{V}$ in graph $\mathcal{G}$, we attempt to explore the attentive correlation between their node representations, i.e., $\mathsf{c}_i \in \boldsymbol{C}$ and $\mathsf{c}_j \in \boldsymbol{C}$, expressed as follows:

$$e(\mathsf{c}_i, \mathsf{c}_j) = a^{\mathsf{c}\top} \text{LeakyReLU}\left(\boldsymbol{W}_{\mathsf{c}}^{e} \cdot \left[\mathsf{c}_i \,\|\, \mathsf{c}_j\right]\right), \tag{28}$$

$$\beta_{ij}^{\mathsf{c}} = \text{Softmax}(e(\mathsf{c}_i, \mathsf{c}_j)) = \frac{\exp(e(\mathsf{c}_i, \mathsf{c}_j))}{\sum_{j' \in \mathcal{A}_i} \exp(e(\mathsf{c}_i, \mathsf{c}_{j'}))}, \tag{29}$$

$$\mathsf{c}_i^{out} = \text{LeakyReLU}\Big(\sum_{j \in \mathcal{A}_i} \beta_{ij}^{\mathsf{c}} \cdot \boldsymbol{W}_{\mathsf{c}} \mathsf{c}_j\Big), \tag{30}$$

where a^{c} is a learnable vector with random initialization. $\boldsymbol{W}_{\mathsf{c}}^{e} \in \mathbb{R}^{d_{\mathsf{c}} \times 2d_{\mathsf{c}}}$ and $\boldsymbol{W}_{\mathsf{c}}^{s} \in \mathbb{R}^{d_{\mathsf{c}} \times d_{\mathsf{c}}}$ are trainable matrices. Herein, we use multi-head GAT in the TFP, where the head number is set to 3. As such, the output can be denoted as $\boldsymbol{C}^{out} \in \mathbb{R}^{n \times 3d_{\mathsf{c}}}$. Then we obtain the final results with the linear projection, denoted as:

$$\widehat{\boldsymbol{X}} = \text{Linear}(\boldsymbol{C}^{out}) \in \mathbb{R}^{n \times \omega}. \tag{31}$$

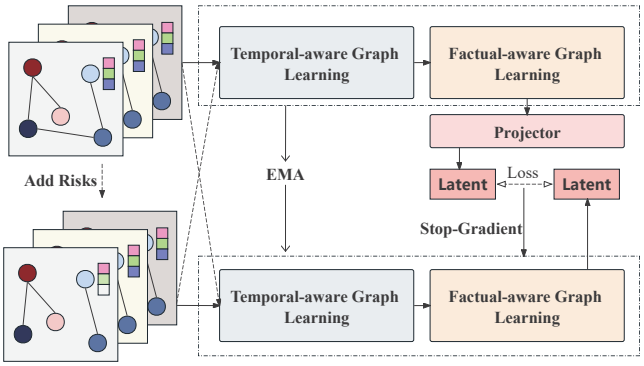


Figure 3: DvBoot mechanism.

3.4 Dual-view Bootstrap (DvBoot)

Technical Bound Investigation: DvBoot treats the above TDL handling the raw traffic data as the deterministic (risk-free) view, denoted as $\mathcal{F}_\theta(\theta \subset \Theta)$, while building a Siamese of TDL which uses different parameter settings to handle multiple potential risks, denoted as the risk-aware view $\mathcal{F}_\kappa$. As shown in Fig. 3, we align these two views by optimizing the mean squared error in the latent space. For instance, given a real (risk-free) flow observations $\boldsymbol{X}$ and its augmented (risk-aware) version $\boldsymbol{X}'$, we have:

$$\mathcal{L}_{\theta,\kappa} = ||\mathcal{H}(\mathcal{F}_\theta(\boldsymbol{X})) - \mathcal{F}_\kappa(\boldsymbol{X}')||_2^2, \tag{32}$$

where $\mathcal{H}$ is a two-layered projection. $\mathcal{H}$ should be nonlinear and contain batch normalization(BN). The BN used in $\mathcal{H}$ could implicitly introduce a negative term [33] which acts as a crucial component to stabilize training. We denote it as:

$$\mathcal{H}(\cdot) \mapsto \text{Linear}(\text{ReLU}(\text{BatchNorm}(\text{Linear}(\cdot)))). \tag{33}$$

Intuitively, we can begin with the optimization by randomly initializing the above parameters θ and κ, respectively. However, this straightforward strategy inevitably leads to a collapse of representations, even useless ones [4, 11, 13]. Besides, maintaining the gradients regarding parameters κ brings additional memory and computation costs. Fortunately, recent exponential moving average (EMA) theory [4, 11, 14, 25, 31] successfully applied in computer vision (CV) suggests that we can establish a conditional gradient update rule between θ and κ (more technical details about EMA are introduced in Appendix A). That is, the update of κ in each training step is *stop-gradient*, which, instead, can be obtained by:

$$\theta \leftarrow \theta - \eta \nabla, \mathcal{L}_{\theta,\kappa} \tag{34}$$

$$\kappa \leftarrow \lambda\kappa + (1-\lambda)\theta, \tag{35}$$

where the optimization of $\mathcal{F}_\kappa$ begins with $\kappa = \theta$ and $\lambda \in [0, 1]$ is a decay rate, η is the learning rate. The gradient of dual-view loss can be obtained by:

$$\begin{aligned} \nabla_\theta \mathbb{E}\left[\|\mathcal{H}^*(z_\theta) - z'_\kappa\|_2^2\right] &= \nabla_\theta \mathbb{E}\left[\|\mathbb{E}[z'_\kappa|z_\theta] - z'_\kappa\|_2^2\right] \\ &= \nabla_\theta \mathbb{E}\left[\sum_i \text{Var}(z'_{\kappa,i}|z_\theta)\right], \end{aligned} \tag{36}$$

where $z_\theta = \mathcal{F}_\theta(\boldsymbol{X})$ and $z'_\kappa = \mathcal{F}_\kappa(\boldsymbol{X}')$. Hence, the optimal objective can be defined as follows:

$$\mathcal{H}^* \triangleq \arg\min_{\mathcal{H}} \mathbb{E}\left[\|\mathcal{H}(z_\theta) - z'_\kappa\|_2^2\right]. \tag{37}$$

Because of the stop-gradient, we can ignore the gradient update of κ, and the gradient of θ can be written in the following form:

$$\frac{\partial}{\partial\theta}\mathbb{E} = \mathbb{E}\left[\frac{\partial L}{\partial \mathcal{H}} \cdot \frac{\partial \mathcal{H}^*}{\partial \theta} + \frac{\partial L}{\partial \mathcal{H}} \cdot \frac{\partial \mathcal{H}^*}{\partial z_\theta} \cdot \frac{\partial z_\theta}{\partial \theta}\right]. \tag{38}$$

It can be hypothesized from the above gradient formula that too slow updating of the parameters of $\mathcal{H}$ during the training will affect the normal gradient propagation process of the model, which requires adjusting the learning rate of $\mathcal{H}$ inconsistent with $\mathcal{F}_\theta$ or adjusting the dimension d_c to control the complexity of $\mathcal{H}$ [11].

Risk Setting: Now we turn to explain how to produce multiple risk-aware versions of an actual (risk-free) view. In our context, we mainly dig into three frequently present risks, including the missing facts, the missing spatial connectivity, and the flow noise, which have been widely posed recently [15, 23, 39, 41]. Specifically, we choose the following strategies: (R1) *Situation Mask*: As each $X \in \mathbb{R}^{n \times \omega}$ ($X \in \boldsymbol{X}$) describes one type of situation records (e.g., the current flow volumes) of n nodes during the time step $t - \omega + 1$ to time step t, we thus mask part of the observed situations conditional on a masking threshold. (R2) *Edge Mask*: we manipulate the spatial connectivity by edge removing, Given the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we obtain an augmented version $\mathcal{E}'$ by randomly removing some of the edges present. (R3) *Flow Noise*: To further stimulate the uncertainty of flow records, we add Gaussian noise to some of the actual flow observations by setting the noise ratio.

Risk Swapping: Inspired by recent bootstrap learning [4, 11, 42], we employ a swapping method to enhance the learning ability of TDL. That is, in the training process, we iteratively swap the risk-free and risk-aware views to TDL $\mathcal{F}_\theta$ and its Siamese $\mathcal{F}_\kappa$, thus the goal of Eq. (32) can be rewritten as follows:

$$\mathcal{L}_{bs} = \frac{1}{2}[||\mathcal{H}(\mathcal{F}_\theta(\boldsymbol{X})) - \mathcal{F}_\kappa(\boldsymbol{X}')||_2^2 + ||\mathcal{H}(\mathcal{F}_\theta(\boldsymbol{X}')) - \mathcal{F}_\kappa(\boldsymbol{X})||_2^2]. \tag{39}$$

3.5 Task Learning

In the end, we summarize the task learning details. In each training epoch, given traffic flow observations $\boldsymbol{X}$ and traffic graph $\mathcal{G}$,

Algorithm 1: The training process of Gboot.

Input: Traffic graph $\mathcal{G}$, the training set of traffic flow $\mathcal{D}_{\boldsymbol{X}}$.

1 Initialize Θ in Gboot;
2 **for** *sample a batch* $\{\boldsymbol{X}_i\}_{i=1}^{B} \in \mathcal{D}_{\boldsymbol{X}}$ **do**
3 **for** $i \in B$ **do**
  // Risk setting for input graph and traffic flow
4 $(\mathcal{G}', \boldsymbol{X}_i') \xleftarrow{risks} (\mathcal{G}, \boldsymbol{X}_i)$;
  // Use TDL for each of the two views
5 $\mathcal{Z}, \mathcal{Z}' = \mathcal{F}_\theta(\boldsymbol{X}_i), \mathcal{F}_\kappa(\boldsymbol{X}_i')$;
  // Swap the input and extract features again
6 $\mathcal{Z}_s, \mathcal{Z}_s' = \mathcal{F}_\theta(\boldsymbol{X}_i'), \mathcal{F}_\kappa(\boldsymbol{X}_i)$;
  // Use TFP module for flow prediction
7 $\widehat{Y} = \mathcal{P}(\mathcal{Z})$;
8 **end**
  // Calculate Bootstrap loss between two views
9 $\mathcal{L}_{bs} = \frac{1}{2}[||\mathcal{H}(\mathcal{Z}) - \mathcal{Z}'||_2^2 + ||\mathcal{H}(\mathcal{Z}_s) - \mathcal{Z}_s'||_2^2]$;
  // Calculate Huber loss
10 $\mathcal{L}_{ta} = \text{HuberLoss}(Y, \widehat{Y})$;
11 $\mathcal{L} = (1-\alpha)\mathcal{L}_{ta} + \alpha\mathcal{L}_{bs}$;
12 update $\mathcal{F}_\theta(\cdot)$, $\mathcal{H}(\cdot)$ and $\mathcal{P}(\cdot)$ to minimize $\mathcal{L}$;
  // Updating parameters with EMA
13 $\kappa \leftarrow \lambda\kappa + (1-\lambda)\theta$;
14 **end**

Output: Obtain the optimal Θ^*.

we treat $\boldsymbol{X}$ as the deterministic (risk-free) view and generate an augmented version $\boldsymbol{X}'$ that contains different risk conditions (i.e., the risk-aware view). Then, we employ TDL and its Siamese to produce the predictive representations regarding these two views, respectively. After that, the output of TDL will be fed into the TFP for flow forecasting. We follow previous studies [21, 30] and employ the Huber loss as the task loss function, which is less sensitive to outliers than the squared error loss, denoted as follows:

$$\mathcal{L}_{ta} = \begin{cases} \frac{1}{2}(Y - \widehat{Y})^2 & \text{for } |Y - \widehat{Y}| \leq \delta, \\ \delta(|Y - \widehat{Y}| - \frac{1}{2}\delta & \text{otherwise.} \end{cases} \tag{40}$$

Herein, δ is a threshold used to control the transition point between square loss and absolute loss. Meanwhile, we produce the bootstrap objective (cf. Eq. (39)). We treat this objective as a weak signal to enhance task learning. In sum, our final objective is denoted as:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{ta} + \alpha\mathcal{L}_{bs}, \tag{41}$$

where α is a hyperparameter for the trade-off between two losses.

Algorithm 1 summarizes the workflow of Gboot. In each epoch, given traffic flow observations $\boldsymbol{X}$ (e.g., $\boldsymbol{X}_i$) and traffic graph $\mathcal{G}$, we treat $\boldsymbol{X}$ as the risk-free view and generate an augmented version $\boldsymbol{X}'$ that contains the uncertain risks (line 4). Then, we employ TDL and its Siamese to respectively produce predictive representations regarding the risk-free view and the risk-aware view (lines 5 and 6) with a swapping strategy. After that, the TDL output will be fed into the TFP for flow forecasting (line 7). Meanwhile, we produce the bootstrap objective (line 9) and the Huber loss (line 10). We treat this objective as a weak signal to enhance task learning (line 11). After model convergence, we employ the optimal Gboot for inference. Notably, TDL employs multiple attention mechanisms (e.g., self-attention and GAT), which can operate in parallel and contribute to efficiency. Moreover, even if we build a Siamese version of TDL, its initial parameters stem from TDL's. Moreover, its optimization is totally dependent on TDL. Therefore, we do not need additional memory to preserve massive gradients relative to this Siamese.

4 Experiments

4.1 Experimental Setup

Datasets: We choose several real-world traffic flow datasets, including four graph-based highway datasets PEMS03, PEMS04, PEMS07, PEMS08 from California [30], and two grid-based datasets, i.e., NYCTaxi [26] and NYCBike [43]. For the four highway datasets, the original traffic flow data has been aggregated into 5-minute intervals and normalized to zero mean, yielding 12 time steps for each hour. We use traffic flow data from the past hour to forecast the flow for the next hour, i.e., $\omega = 12$ in alignment with previous studies [21, 23, 30]. For the taxi dataset dataset, the original traffic flow data has been aggregated into 30-minute intervals and normalized to zero mean. For the bike dataset dataset, the original traffic flow data has been aggregated into 60-minute intervals and normalized to zero mean. We use traffic flow data from the past 12 time steps to forecast the flow for the next 12 time steps. Statistics of all datasets is summarized in Table 1. In addition, we follow the standard dataset split manner by dividing the original traffic data into training, validation and testing sets with ratio 6:2:2.

Table 1: The statistics of used datasets.

Dataset	#Node	#Edge	#Time step	Time Span
PEMS03	358	547	26208	9/1/2018–11/30/2018
PEMS04	307	340	16992	1/1/2018–2/28/2018
PEMS07	883	866	28224	5/1/2017–8/31/2017
PEMS08	170	295	17856	7/1/2016–8/31/2016
NYCTaxi	200	/	17520	1/1/2014–12/31/2014
NYCBike	128	/	4392	4/1/2014-9/30/2014

Baselines: We compare with the following 12 representative baselines: *DCRNN* [24] uses diffusion graph convolutional networks and seq2seq to explore spatial and temporal dynamics, respectively. *ASTGCN (r)* [12] is an attention-based spatiotemporal GCN that contains a spatial attention network and a temporal attention network. *STGCN* [40] uses spatial-graph convolution and temporal-gated convolution to capture spatial and temporal dependencies, respectively. *STSGCN* [30] utilizes local spatial-temporal subgraph modules to explore spatial and temporal dependencies synchronously. *STFGNN* [22] develops a spatial-temporal fusion graph to compensate for existing spatial correlations. *AGCRN* [2] advances GCNs with node embeddings to enhance node-specific spatial and temporal correlations in traffic series. *STGODE* [8] extends multiple GCNs with a tensor-based ODE and utilizes a convolution layer to capture temporal dependencies. *DSTAGNN* [21] operates attention-based methods to explore spatial and temporal dependencies while employing gated convolutions to explore different ranges of temporal dependencies. We select its variants *DSTAGNN-G* relying on real-world spatial connectivity for fairness. *SPGCL* [23] involves contrastive learning with three graph-based augmentations to enhance informative relations. *FourierGNN* [38] introduces the Fourier Graph Operator to perform matrix multiplications in Fourier space and make multivariate time series forecasting from a pure graph

Table 2: Performance comparison of Gboot and baselines on PEMS03, PEMS04, PEMS07, PEMS08.

Model	PEMS03			PEMS04			PEMS07			PEMS08		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	18.18	30.31	18.91%	24.70	38.12	17.12%	25.30	38.58	11.66%	17.86	27.83	11.45%
ASTGCN(r)	17.69	29.66	19.40%	22.93	35.22	16.56%	28.05	42.57	13.92%	18.61	28.16	13.08%
STGCN	17.49	30.12	17.15%	22.70	35.55	14.59%	25.38	38.78	11.08%	18.02	27.83	11.40%
STSGCN	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	16.77	28.34	16.30%	20.48	32.51	16.77%	23.46	36.60	9.21%	16.94	26.25	10.60%
AGCRN	15.98	28.25	15.23%	19.83	32.26	12.97%	22.37	36.55	9.12%	15.95	25.22	10.09%
STGODE	16.50	27.84	16.69%	20.84	32.82	13.77%	22.59	37.54	10.14%	16.81	25.97	10.62%
DSTAGNN-G	15.61	27.23	14.79%	19.41	31.63	12.84%	21.67	35.04	9.06%	15.90	25.24	9.97%
SPGCL	23.31	37.37	21.88%	24.75	40.12	16.34%	31.35	46.34	18.32%	19.92	33.68	15.77%
FourierGNN	17.27	27.20	15.88%	22.98	36.23	15.14%	25.47	39.69	10.76%	18.14	28.39	11.35%
GraphWaveNet	19.12	32.77	18.89%	24.89	39.66	17.29%	26.39	41.50	11.97%	18.28	30.05	12.15%
PDFormer	21.82	36.75	21.47%	25.75	42.09	17.55%	23.92	36.76	11.62%	20.30	33.26	12.54%
Gboot	**15.43**	**26.42**	**14.51%**	**19.28**	**31.02**	**12.58%**	**21.35**	**34.43**	**9.02%**	**15.54**	**24.53**	**9.76%**

Table 3: Performance comparison of Gboot and baselines on NYCTaxi, NYCBike.

Model	NYCTaxi Inflow			NYCTaxi Outflow			NYCBike Inflow			NYCBike Outflow		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	4.77	6.61	27.17%	9.40	13.36	37.70%	12.53	18.07	39.76%	11.96	16.89	41.54%
ASTGCN(r)	4.67	6.26	28.06%	9.76	13.74	37.35%	6.02	11.15	18.31%	5.72	10.96	16.93%
STGCN	5.48	7.45	31.35%	11.83	16.41	43.25%	7.44	12.13	23.48%	5.53	10.56	17.44%
STSGCN	5.01	6.31	32.83%	9.67	13.24	39.58%	5.63	11.04	16.75%	5.51	10.51	17.39%
STFGNN	4.85	6.29	31.13%	9.65	13.11	42.27%	6.06	10.93	18.29%	6.42	10.97	21.42%
AGCRN	11.14	14.31	62.77%	18.05	22.47	69.82%	7.37	11.92	24.60%	6.19	10.87	20.36%
STGODE	5.89	7.53	40.50%	9.71	13.15	50.41%	5.93	10.99	17.88%	5.44	10.25	17.53%
DSTAGNN-G	5.69	7.94	28.80%	9.60	12.73	44.06%	6.05	10.89	19.09%	7.06	11.86	22.64%
SPGCL	4.76	6.56	26.92%	10.58	14.29	41.59%	11.44	17.12	33.21%	11.02	16.21	34.59%
FourierGNN	6.08	8.33	34.32%	10.85	14.55	44.72%	6.59	11.88	19.97%	6.22	11.94	18.68%
GraphWaveNet	4.69	6.28	29.55%	10.45	14.08	43.26%	7.93	13.63	25.40%	7.05	12.44	22.31%
PDFormer	4.55	6.35	25.39%	9.99	12.88	44.51%	6.42	11.31	20.08%	6.89	11.96	21.58%
Gboot	**4.44**	**6.18**	**24.84%**	**9.27**	**12.63**	**37.03%**	**5.33**	**10.51**	**15.61%**	**5.21**	**10.08**	**16.50%**

perspective. *Graph WaveNet* [36] develops a novel adaptive dependency matrix and a stacked dilated 1D convolution component to capture the hidden spatial-temporal dependency in the data. *PDFormer* [16] proposes a novel propagation delay-aware dynamic long-range transformer for accurate traffic flow prediction.

Implementations: Gboot is implemented with PyTorch and uses one NVIDIA RTX 4090 GPU. In Gboot, the head number H is set to 3. The dimensions d_t and d_f are set to 32. The filter sizes for Temporal Latent Capture and Factual Latent Capture are set to $\{1\times3, 1\times5, 1\times7\}$. d_c in TDL is 128, d_S in Temporal-Spatial Aggregation and d_P in Factual-Spatial Aggregation are both 256. We optimize with the Adam optimizer for a maximum of 50 epochs. The batch size is 32 and the initial learning rate is 0.003. For reproducibility, the source codes are available at https://github.com/wangzz-yyzz/Gboot.

Metrics: We follow existing studies [23, 24, 30] and evaluate performance by three common metrics, including mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE). We filter the samples with flow values below 10 when evaluating the NYCTaxi and NYCBike, consistent with [37].

4.2 Main Results

Performance Comparison. Table 2 reports the performance results of our proposed Gboot and the baselines on four graph-based datasets. Additionally, Table 3 reports the performance results on two grid-based datasets. We can observe that our Gboot consistently outperforms the baselines, demonstrating the effectiveness of the proposed solution. Among the baselines, solutions such as DCRNN and STGCN that explore spatial and temporal dependencies, respectively, perform poorly, suggesting that insufficient dependencies may hinder the capture of implicit patterns behind flow dynamics. Methods such as STSGCN that take into account spatio-temporal correlations exhibit better performance, implying that incorporating spatio-temporal interactions can indeed facilitate the accurate capture of flow patterns. STGODE and DSTAGNN-G operated temporal convolution to explore long-range temporal dependencies with promising results, which indicates that using convolutional operations for temporal dependency learning is a useful alternative because it can flexibly tackle different ranges of temporal dynamics with filter settings. SPGCL is a contrastive learning solution for handling uncertain risks, akin to the risk settings used in our practice – but, surprisingly, we find that it performs poorly. We consider the reason for this to be that it crafts massive negative instances to enhance the disentanglement of graph node representations, resulting in instability in graph learning. In short, Gboot performs the best, and we conjecture that the reason is two-fold. First, it considers both temporal and factual interactive dynamics over spatial dependency learning, providing us with more prior knowledge about flow trends. Second, it provides a simple dual-view bootstrap mechanism to boost dependency dynamic learning, enhancing its robustness by handling different potential uncertain risks.

Efficiency Comparison. Table 4 reports the efficiency comparison on PEMS04 and shows that Gboot does not introduce much computational time in the inference phase as multiple attention mechanisms can operate in parallel. The main computational overhead comes from the dual-view design in the training phase.

Table 4: Efficiency evaluation on PEMS04.

Model	Train(s/epoch)	Inference(s)
AGCRN	40.9	3.8
DSTAGNN-G	84.9	6.1
FourierGNN	20.1	2.3
GraphWaveNet	231.5	26.5
PDFormer	73.8	9.1
Gboot (ours)	**31.8**	**2.4**

4.3 Ablation Study

We conduct two groups of ablation studies, including:

Block Design. To evaluate the effectiveness of the modules design, we tested multiple variants of Gboot: *w/o Proj* (removes the UTP), *w/o GAT* (removes the GAT in the TFP), *w/o DV* (removes the DvBoot), *w/o HF* (removes the situations regarding historical flow, i.e., $c = 1$), *w/o TGL* (removes the TGL block) and *w/o FGL* (removes the FGL block). Fig. 4 shows that removing each block results in performance degradation, indicating the effectiveness of each designed block. Moreover, we observe that removing the TGL block that explores temporal dependencies over spatial connectivity learning has the most significant impact on the model performance, uncovering the fact that most of the studies have verified that considering temporal dependencies is indeed crucial for perceiving future flow trends. The results of *w/o TGL* and *w/o HF* demonstrate that considering the factual interaction between different situations can boost the ability of model forecasting. Finally, *w/o DV* performance suggests that considering the uncertain risks can enhance the generalization of high-accuracy traffic trends.

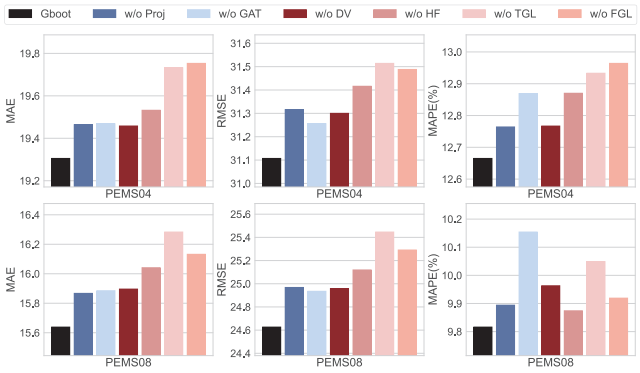


Figure 4: The impact of module design in Gboot.

Risk Setting. We investigate the impacts of different risks by removing each of the risk settings (i.e., *w/o R1, w/o R2*, and *w/o R3*) and using each of the risk settings (i.e., *only R1*, *only R2* and *only R3*). As shown in Fig. 5, among the 'w/o' series, we can observe that removing the *Edge Mask* shows the worst results (cf. *w/o R2*), which indicates that incorporating the uncertainty of graph topology can enhance the model performance. Urban areas often have road closures and the like, and it becomes very practical to consider this risk condition. Among the 'only' series, we can observe that only using one of them does not promote the model's ability well. We consider the plausible reason is that inadequate insight into risk setting would hinder the learning of dual-view bootstrap.

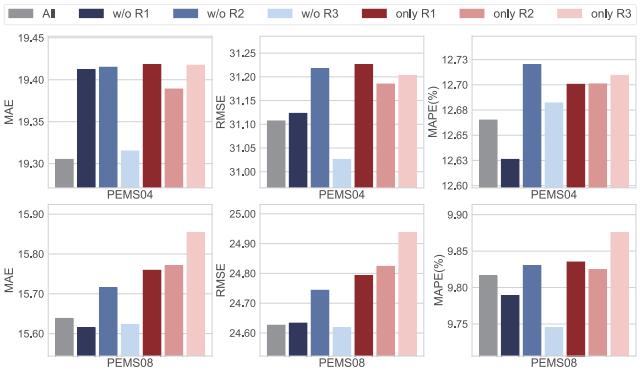


Figure 5: The impact of risk settings.

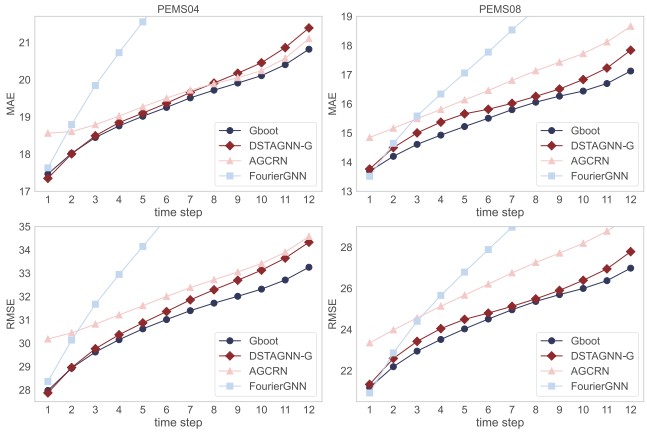
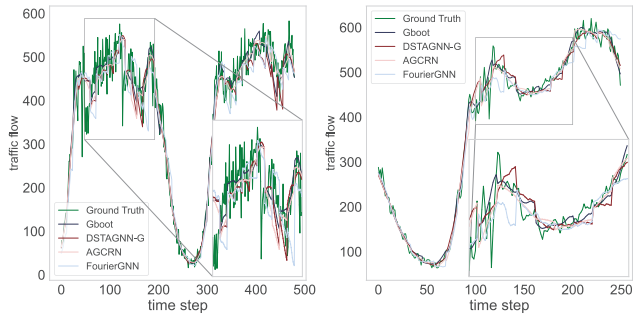


Figure 6: Performance comparison in per forecasting step.

4.4 Forecasting Investigation

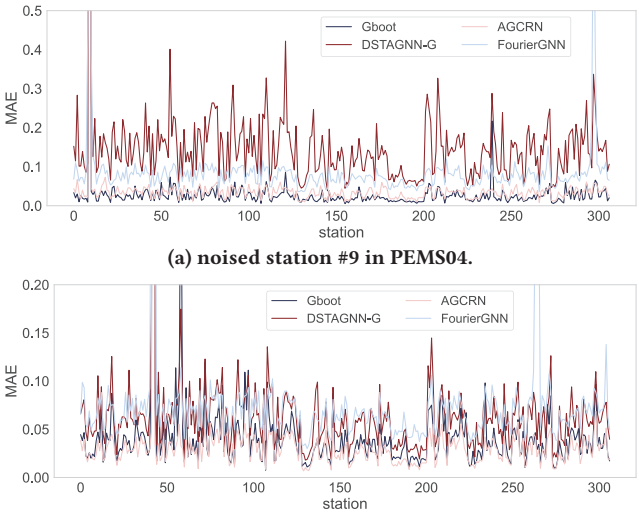
Long-range Forecasting. To show the model performance in long-range forecasting, we compute the error (i.e., MAE or RMSE) for each of the predicted 12 time steps individually. Fig. 6 displays the measurements of MAE and RMSE on PEMS04 and PEMS08. We can find that as the forecasting step increases, the tendency for our Gboot to have lower errors on both metrics becomes more prominent than other recently representative baselines. Moreover, we present visualizations comparing the longer-term forecasting results against the ground truth using the testing data from PEMS04 and PEMS08, respectively. Fig. 7 shows a snapshot of the testing data – we mark representative methods and ours with different colors. We note that, due to space limits, an enlarged version of Fig. 7 is provided in Appendix B (cf. Fig. 13). We find that the ground truth is highly volatile. However, Gboot is remarkably more proximate to the ground truth (cf. zoomed parts) while the other baselines deviate more significantly. This suggests that Gboot can learn better traffic flow regularities to predict longer-term trends.

Noise Immunity Test. As we claimed, Gboot has the comprehension ability to tackle uncertainty or noise. We conduct a noise immunity test by respectively noising a station's traffic flow. Specifically, we make comparative experiments in which one group utilized raw historical flow observations and the other group utilized violently noisy observations, i.e., Gaussian noise was introduced into each



(a) station #122. **(b) station #80.**

Figure 7: Comparison of forecasting curves.



(a) noised station #9 in PEMS04.

(b) noised station #42 in PEMS08.

Figure 8: A noise Immunity test.

observation. We have this hypothesis: *If the model exhibits strong noise immunity, the disparities between the two groups of comparative forecasts are expected to converge.* To confirm this hypothesis, we adopt the MAE metric to calculate the difference between the two groups. As shown in Fig. 8, the x-axis refers to the different stations while the y-axis shows the MAE gap between the two groups. We can find that Gboot consistently owns the lowest MAE on each station, suggesting it has the most robust noise immunity.

Spatial Connectivity Analysis. We investigate station (sensor)-aspect forecasting errors to uncover the impact of spatial connectivity. In Fig 9, we calculate MAE values for all forecasting time steps at all stations with the different number of neighbors and plot the frequency distribution of occurrences, where *less* refers to the station only has one neighbor while the *more* means the station has more than 5 spatial neighbors. We find that the peaks of $\text{Gboot}_{\text{more}}$ are distributed in the small error region or its peaks are lower than those of $\text{Gboot}_{\text{less}}$, suggesting that exposing the spatial connectivity do affect the forecasting performance. In fact, aggregating informative knowledge from the spatial neighbors could bring uncertain noise. Gboot can tackle this issue well, which shows its robustness in handling information aggregation behind spatial connectivity.

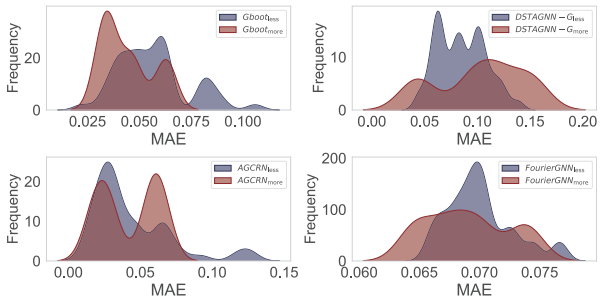


Figure 9: The investigation on spatial connectivity.

Sensitivity Analysis. We present a case investigation here due to space limitations. Fig. 10 shows that a large masking ratio in situation masking (risk setting R1) degrades the model's performance. However, if the masking ratio is set too low, it fails to effectively introduce the desired uncertain risks. Notably, a more comprehensive sensitivity analysis is provided in Appendix B (cf. Fig. 12).

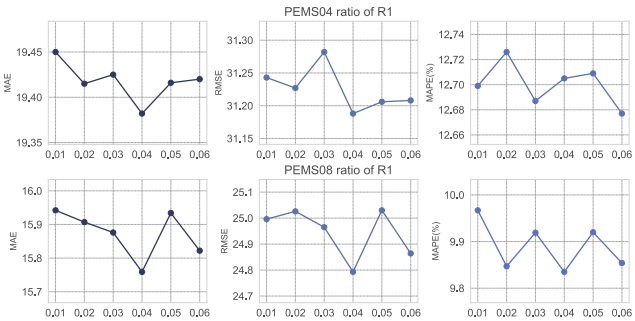


Figure 10: Sensitivity analysis on risk R1.

5 Conclusion

In this study, we identified two challenges that have never been addressed before, i.e., factual interaction and robustness risk. Correspondingly, we introduced a novel solution called Gboot to handle traffic flow forecasting by considering the above challenges. Specifically, Gboot containing a TDL module can capture temporal and factual interactive dynamics on spatial connectivity learning. Moreover, multi-scale gated convolution was applied to each of them in order to explore different ranges of dynamic dependencies. To account for multiple uncertainty risks, we propose a dual-view bootstrap mechanism to improve the generalization of the model, which does not need to involve huge negative instances for comparisons. Finally, our experiments conducted on several representative traffic datasets demonstrate the superiority of the Gboot against the baselines. As part of our future work, we plan to investigate more background context, e.g., point-of-interest distribution, remote sensing semantics, etc., and develop a multimodal solution.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No.62102326, the Natural Science Foundation of Sichuan Province under Grant No.2023NSFSC1411, Sichuan Science and Technology Program under Grant No.2023ZYD0145.

References

[1] Lei Bai, Lina Yao, Salil S Kanhere, Xianzhi Wang, and Quan Z Sheng. 2019. STG2seq: spatial-temporal graph to sequence model for multi-step passenger demand forecasting. In *Proceedings of the 28th IJCAI*. 1981–1987.

[2] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems* 33 (2020), 17804–17815.

[3] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks?. In *International Conference on Learning Representations*.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9650–9660.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[6] Weiqi Chen, Ling Chen, Yu Xie, Wei Cao, Yusong Gao, and Xiaojie Feng. 2020. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In *AAAI conference on artificial intelligence*, Vol. 34. 3529–3536.

[7] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International conference on machine learning*. PMLR, 933–941.

[8] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 364–373.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*. PMLR, 1243–1252.

[10] Zili Geng, Jie Xu, Rongsen Wu, Changming Zhao, Jin Wang, Yunji Li, and Chenlin Zhang. 2024. STGAFormer: Spatial–temporal Gated Attention Transformer based Graph Neural Network for traffic flow forecasting. *Information Fusion* (2024), 102228.

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems* 33 (2020), 21271–21284.

[12] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI conference on artificial intelligence*, Vol. 33. 922–929.

[13] Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. 2020. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3875–3886.

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[15] Jiahao Ji, Jingyuan Wang, Chao Huang, Junjie Wu, Boren Xu, Zhenhe Wu, Junbo Zhang, and Yu Zheng. 2023. Spatio-temporal self-supervised learning for traffic flow prediction. In *AAAI conference on artificial intelligence*, Vol. 37. 4356–4364.

[16] Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. 2023. Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. In *AAAI conference on artificial intelligence*, Vol. 37. 4365–4373.

[17] Renhe Jiang, Zhaonan Wang, Jiawei Yong, Puneet Jeph, Quanjun Chen, Yasumasa Kobayashi, Xuan Song, Shintaro Fukushima, and Toyotaro Suzumura. 2023. Spatio-temporal meta-graph learning for traffic forecasting. In *AAAI conference on artificial intelligence*, Vol. 37. 8078–8086.

[18] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* 207 (2022), 117921.

[19] Yilun Jin, Kai Chen, and Qiang Yang. 2023. Transferable Graph Structure Learning for Graph-based Traffic Forecasting Across Cities. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1032–1043.

[20] S Vasantha Kumar and Lelitha Vanajakshi. 2015. Short-term traffic flow prediction using seasonal ARIMA model with limited input data. *European Transport Research Review* 7, 3 (2015), 1–9.

[21] Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. 2022. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *International conference on machine learning*. PMLR, 11906–11917.

[22] Mengzhang Li and Zhanxing Zhu. 2021. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *AAAI conference on artificial intelligence*, Vol. 35. 4189–4196.

[23] Rongfan Li, Ting Zhong, Xinke Jiang, Goce Trajcevski, Jin Wu, and Fan Zhou. 2022. Mining spatio-temporal relations via self-paced graph contrastive learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 936–944.

[24] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[26] Dachuan Liu, Jin Wang, Shuo Shang, and Peng Han. 2022. Msdr: Multi-step dependency relation networks for spatial temporal forecasting. In *28th ACM SIGKDD conference on knowledge discovery and data mining*. 1042–1050.

[27] Yi-Ju Lu and Cheng-Te Li. 2020. Agstn: Learning attention-adjusted graph spatio-temporal networks for short-term urban sensor value forecasting. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1148–1153.

[28] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. 2015. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies* 54 (2015), 187–197.

[29] Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202* (2020).

[30] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. 2020. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *AAAI conference on artificial intelligence*, Vol. 34. 914–921.

[31] Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *31st International Conference on NIPS*. 1195–1204.

[32] David Alexander Tedjopurnomo, Zhifeng Bao, Baihua Zheng, Farhana Murtaza Choudhury, and Alex Kai Qin. 2020. A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering* 34, 4 (2020), 1544–1561.

[33] Yuandong Tian, Lantao Yu, Xinlei Chen, and Surya Ganguli. 2020. Understanding self-supervised learning with dual deep networks. *arXiv preprint:2010.00578* (2020).

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, Vol. 30. 6000–6010.

[35] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic Flow Prediction via Spatial Temporal Graph Neural Network. In *Proceedings of The Web Conference 2020 (WWW '20)*. 1082–1092.

[36] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the 28th IJCAI*. 1907–1913.

[37] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *AAAI conference on artificial intelligence*, Vol. 32.

[38] Kun Yi, Qi Zhang, Wei Fan, Hui He, Liang Hu, Pengyang Wang, Ning An, Longbing Cao, and Zhendong Niu. 2023. FourierGNN: Rethinking Multivariate Time Series Forecasting from a Pure Graph Perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[39] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[40] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th IJCAI*. 3634–3640.

[41] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1294–1303.

[42] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*. PMLR, 12310–12320.

[43] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI conference on artificial intelligence*, Vol. 31.

[44] Qianru Zhang, Chao Huang, Lianghao Xia, Zheng Wang, Siu Ming Yiu, and Ruihua Han. 2023. Spatial-temporal graph learning with adversarial contrastive adaptation. In *International Conference on Machine Learning*. PMLR, 41151–41163.

[45] Xiyue Zhang, Chao Huang, Yong Xu, Lianghao Xia, Peng Dai, Liefeng Bo, Junbo Zhang, and Yu Zheng. 2021. Traffic flow forecasting with spatial-temporal graph diffusion network. In *AAAI conference on artificial intelligence*, Vol. 35. 15008–15015.

[46] Chuanpan Zheng, Xiaoliang Fan, Shirui Pan, Haibing Jin, Zhaopeng Peng, Zonghan Wu, Cheng Wang, and S Yu Philip. 2023. Spatio-temporal joint graph convolutional networks for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[47] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *AAAI conference on artificial intelligence*, Vol. 34. 1234–1241.

Appendix

We now present additional details regarding the derivation of EMA and extended sensitivity analysis.

A The Derivation of EMA

EMA is a widely used practice in recent deep semi-supervised learning and self-supervised learning by following the teacher-student manner. For instance, in our context, we have:

$$\mathcal{L}_{\theta,\kappa} = ||\mathcal{H}(\mathcal{F}_\theta(\boldsymbol{X})) - \mathcal{F}_\kappa(\boldsymbol{X}')||_2^2, \tag{42}$$

where the parameter optimization is followed by the rule:

$$\theta \leftarrow \theta - \eta\nabla\mathcal{L}_{\theta,\kappa} \tag{43}$$

$$\kappa \leftarrow \lambda\kappa + (1-\lambda)\theta. \tag{44}$$

Herein, $\mathcal{F}_\theta$ can be treated as a student network and $\mathcal{F}_\kappa$ as a teacher network with the same network architecture. According [4, 11, 31], in common sense, we expect to enforce the outputs of $\boldsymbol{C} = \mathcal{F}_\theta(\boldsymbol{X})$ and $\boldsymbol{C}' = \mathcal{F}_\kappa(\boldsymbol{X}')$ to be closer in the latent space with the goal of:

$$\mathbb{E}\left[-\log\frac{\exp<\boldsymbol{C},\boldsymbol{C}'>}{\sum_j(\exp<\boldsymbol{C},\boldsymbol{C}'>+\exp<\boldsymbol{C},\boldsymbol{C}'^-_j>)}\right] \tag{45}$$

$$\Rightarrow \mathbb{E}[-<\boldsymbol{C},\boldsymbol{C}'>] + \mathbb{E}[\log\sum_j(\exp<\boldsymbol{C},\boldsymbol{C}'^-_j>)]. \tag{46}$$

The first objective is usually referred to as *alignment*, which is actually Eq. (42). For the second term, it is the *uniformity* objective to enlarge the latent space distance between $\boldsymbol{C}$ and $\boldsymbol{C}'^-_j$. Typically, a group of $\boldsymbol{C}'^-_j$ in self-supervised learning refers to a set of negative samples regarding $\boldsymbol{C}$. Without the second term, $\mathcal{F}_\theta$ and $\mathcal{F}_\kappa$ will be homogenized, resulting in collapse of the latent representations. Hence, in the optimization process, $\mathcal{F}_\kappa$ should have a different but similar representational capacity compared to $\mathcal{F}_\theta$. Recall Eq. (44), the EMA theory provides a view that κ is conditioned on θ but using a decayed optimization manner to maintain the ability of optimization difference, i.e., latency optimization. Hence, recent studies of the teacher-student framework remove the second term while using EMA to maintain the disentanglement ability of teacher network, which does not need to involve unstable negative samples.

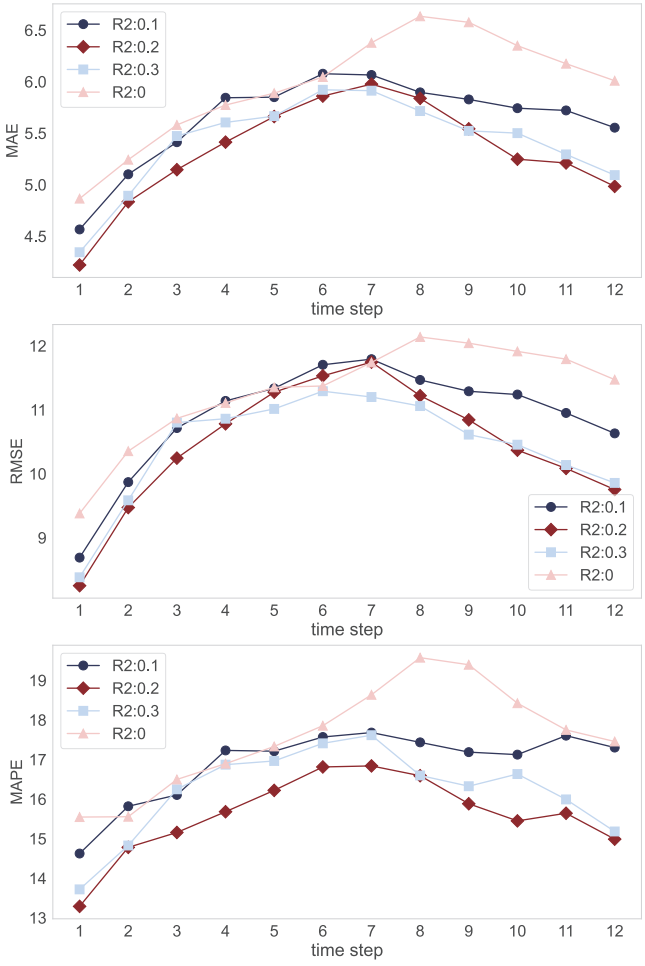


Figure 11: Performance comparison in per forecasting step on NYCBike dataset.

B Additional Sensitivity Analysis

In addition to the short version of the sensitivity analysis in the main text (cf. Sec. 4.4), we provide a more comprehensive analysis of significant hyperparameters that could be sensitive to the model.

Architecture Sensitivity. We first investigate the hyperparameter settings of network architecture, which typically affect the model scale and could affect the model performance. Specifically, we vary the key hyperparameters in the Gboot architecture on PEMS04 dataset, yielding different groups of results. As shown in Table 5, we in general find that they are not extremely sensitive to the model performance, which was also uncovered by previous studies. Considering the trade-off between model scale and performance, our default configuration of Gboot in this study is [32,3,256,128].

Environmental Sensitivity. In this part, we investigate the impact of factor settings, including trade-off factor α in the final objective cf. Eq. (41) and the ratios in risk settings (R1, R2, and R3). The results are reported in Fig. 11 and Fig. 12. We have following observations: *(1) The impact of α.* α is to adjust the contribution between task learning and bootstrap learning. We observe that using either a too large (affecting the model convergence) or a too small (affecting the distillation about uncertain risks) α value will degrade the forecast performance. Hence, this study empirically set α to 0.3.

Table 5: The sensitivity analysis on network architecture.

$d_t/d_f, H, d_S/d_P, d_c$	MAE	RMSE	MAPE
[32,3,256,128]	19.28	31.02	12.58%
[64, 3, 256, 128]	19.49	31.37	12.68%
[16, 3, 256, 128]	19.57	31.38	12.82%
[32, 1, 256, 128]	19.58	31.40	12.82%
[32, 5, 256, 128]	19.38	31.18	12.71%
[32, 3, 128, 128]	19.51	31.30	12.78%
[32, 3, 512, 128]	19.35	31.12	12.69%
[32, 3, 256, 64]	19.45	31.21	12.83%
[32, 3, 256, 256]	19.48	31.19	12.75%

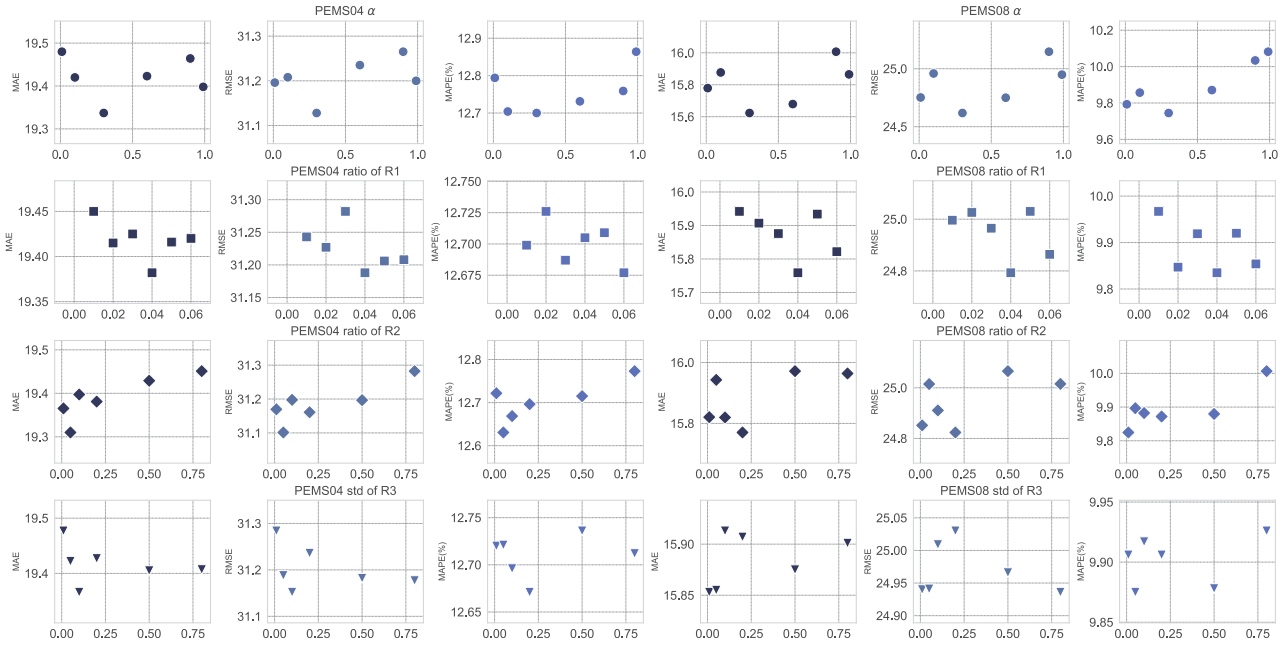
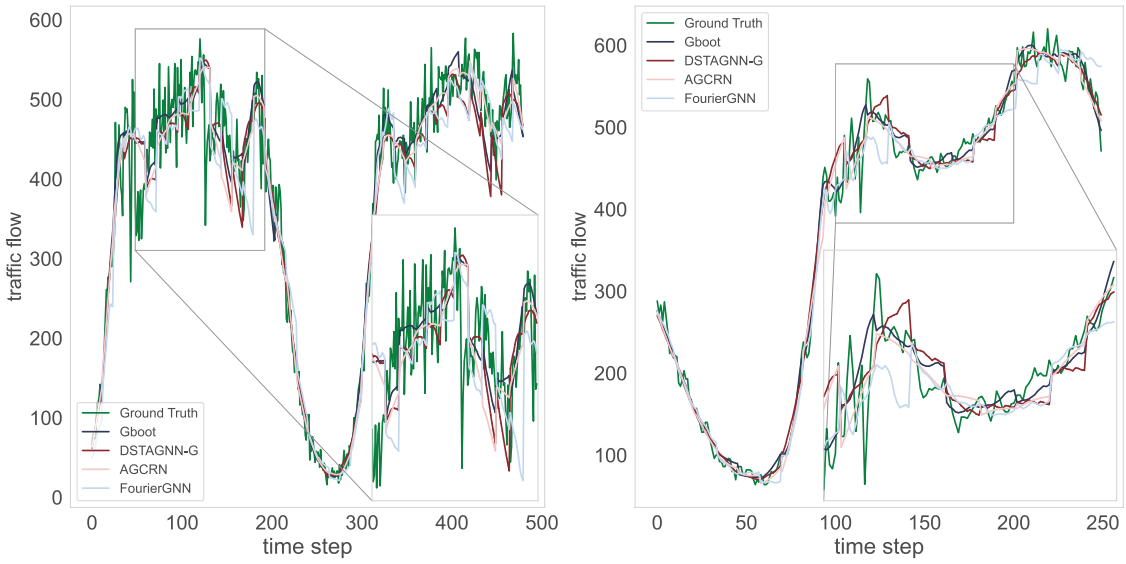


Figure 12: Sensitivity analysis.



(a) station #122 **(b) station #80**

Figure 13: Comparison of forecasting curves.

(2) Risk ratio in R1. This ratio will affect the capture of situational dynamics behind the traffic flows. We observe that too much masking generally leads to poor performance and unstable training. Hence, in this study, we set the mask ratio of R1 to 0.04.

(3) Risk ratio in R2. This ratio will affect the connectivity of the build graph. Obviously, a small value provides a better trade-off between node-useful connectivity and relationship redundancy. In addition, we also find that increasing the proportion of R2 improves the prediction accuracy of the model over long time steps, which can be demonstrated by Fig. 11.

(4) Risk ratio in R3. This factor is to control the noise level of actual flow volumes. We find that adding noise to the actual flows indeed enhances the forecast performance and model robustness. In practice, however, using too many noises will affect model convergence.

Note. We re-iterate here that Fig. 13 is an enlarged version of Fig. 7, as mentioned in the main text.