2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00125

Self-Supervised Spatial-Temporal Bottleneck Attentive Network for Efficient Long-term Traffic Forecasting

Shengnan Guo[1, 2], Youfang Lin[1, 2], Letian Gong[1], Chenyu Wang[1], Zeyu Zhou[1], Zekai Shen[1], Yiheng Huang[1], Huaiyu Wan[1, 2*]

[1] *School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China*

[2] *Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing, China*

{guoshn, yflin, gonglt, wangchenyu, zeyuzhou, zkshen, huangyh, hywan}@bjtu.edu.cn

***Abstract*—In intelligent transportation systems, accurate long-term traffic forecasting is informative for administrators and travelers to make wise decisions in advance. Recently proposed spatial-temporal forecasting models perform well for short-term traffic forecasting, but two challenges hinder their applications for long-term forecasting in practice. Firstly, existing traffic forecasting models do not have satisfactory scalability on effectiveness and efficiency, *i.e.*, as the prediction time spans extend, existing models either cannot capture the long-term spatial-temporal dynamics of traffic data or equip global receptive fields at the cost of quadratic computational complexity. Secondly, the dilemma between the models' strong appetite for high-quality training data and their generalization ability is also a challenge we have to face. Thus how to improve data utilization efficiency deserves thoughtful thinking. Aiming at solving the long-term traffic forecasting problem and facilitating the deployment of traffic forecasting models in practice, this paper proposes an efficient and effective Self-supervised Spatial-Temporal Bottleneck Attentive Network (SSTBAN). Specifically, SSTBAN follows a multi-task framework by incorporating a self-supervised learner to produce robust latent representations for historical traffic data, so as to improve its generalization performance and robustness for forecasting. Besides, we design a spatial-temporal bottleneck attention mechanism, reducing the computational complexity meanwhile encoding global spatial-temporal dynamics. Extensive experiments on real-world long-term traffic forecasting tasks, including traffic speed forecasting and traffic flow forecasting under nine scenarios, demonstrate that SSTBAN not only achieves the overall best performance but also has good computation efficiency and data utilization efficiency.**

***Index Terms*—traffic forecasting, spatial-temporal graph data, self-supervised learning**

I. Introduction

Accurate traffic forecasting is essential for modern intelligent transportation systems (ITS). In general, traffic forecasting can be categorized into short-term forecasting and long-term forecasting [1]–[3]. If the forecasting time span is shorter than an hour, the problem is considered as short-term forecasting; Otherwise, it is considered as long-term forecasting. Compared to short-term forecasting which helps for timely decisions, long-term forecasting provides necessary supporting information for travelers and administrators to optimize trip planning and transportation resource management. In particular, traffic forecasting information covering a few hours in the future is helpful for users to make routing plans in advance. Daily traffic forecasting is crucial for traffic flow guidance and fixed-time signal control strategies. In fixed-time signal control [4], traffic on the target day needs to be forecast first, then the whole day is split into several intervals accordingly, and finally administrators make an appropriate signal timing plan for each interval.

Recently, many efforts have been made to advance the technologies for short-term traffic forecasting. All of them prove that deep-learning based methods, especially the Spatial-Temporal Graph Neural Networks (STGNNs) [5] are more capable than the classic time series analysis models and the traditional machine learning models when solving the short-term traffic forecasting task. Specifically, STGNNs regard traffic networks as graphs, in which nodes refer to the monitor stations, employ graph neural networks to capture the dynamics of traffic data over the spatial dimension, and adopt forward computation functions to model the dynamics across the temporal dimension. Despite of its great success in short-term traffic forecasting, when the existing STGNNs are applied for long-term traffic forecasting in practice, two tricky problems occur.

Firstly, the algorithm's effectiveness and efficiency become one of the most concerns when making long-term traffic forecasting. As compared to the short-term traffic forecasting, when making long-term forecasting, we need to feed more time slices of historical traffic as input to capture the global changing trend of traffic data meanwhile generating traffic data in much more future time slices. Thus, long-term traffic forecasting requires the algorithms to have the ability to capture the long-term dynamics (the definition of dynamics has been given in [6]) within each monitor stations and those among monitor stations in an efficient and effective manner. However, existing STGNNs are either unable to capture the global dynamic correlations of traffic series or computational costly, which makes it impractical to deploy and train them in ITS. Specifically, according to the choice of

Corresponding author: hywan@bjtu.edu.cn

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00125

forward computation function, STGNNs can be divided into RNN-based, CNN-based and attention-based approaches [7]. The gradient vanishing issue in RNNs makes it hard for RNN-based approaches to capture long-term dynamics. In addition, the computations in RNNs must be done sequentially, so the training time of RNN-based approaches linearly increases with the forecasting time span extending. By contrast, although the training process of CNN-based approaches can be sped up by parallel computing, the local kernel size in CNNs limits their ability to capture long-term temporal dynamics. Compared with them, attention-based approaches are more flexible. They are able to model the dynamics of traffic data without regard to the distances between elements along the spatial and temporal dimensions. But attention-based models have quadratic computational complexity, so their training is prohibitively costly with regard to memory, especially on long sequences or large traffic networks.

Secondly, in addition to algorithm efficiency, data utilization efficiency is another practical dilemma we have to face in ITS. Here, the term "data utilization efficiency" refers to the efficiency of the prediction algorithm in utilizing data, *i.e.*, whether or not the prediction algorithm leads to the desired outcome within data resource constraints. Improving data utilization efficiency means enhancing the prediction algorithm's generalization ability meanwhile relaxing its excessive requirement on the quantity and quality of training data since storing and pre-processing data increases the costs of ITS. However, existing widely used STGNNs usually have a strong appetite for high-quality data. They suffer from over-fitting and learning spurious correlations when training data is limited and noisy, resulting in unsatisfactory generalization performance. While few studies pay attention to this practical problem.

Fortunately, we could find feasible solutions and valuable ideas in other fields on how to improve the data efficiency of high-capacity deep-learning models. Specifically, in natural language processing (NLP) and computer vision (CV), self-supervision learning has been proven to be an effective way to improve data efficiency [8]. For example, BERT [9] in NLP and MAE [10] in CV adopt self-supervised learners through pre-training to leverage the inherent co-occurrence dependencies of data so that they can capture the underlying general and universal patterns of sentences and images, respectively. Consequently, with the help of self-supervised pre-training, using limited training data to train high-capacity models that generalize well becomes possible. These successes inspire us to wonder *"Can we adopt self-supervised learners to improve the data utilization efficiency of STGNNs for the traffic forecasting task"*.

Following this idea, in order to correctly incorporate self-supervised learning into STGNNs to improve data utilization efficiency, we need to realize the differences between the tasks in NLP/CV and our spatio-temporal traffic forecasting task. In NLP/CV, fundamental patterns like the shapes of objects in images or the semantic meanings of the words in sentences are universal across datasets. So studies in NLP and CV incorporate self-supervised learning by the pre-training paradigm so that they can utilize the universal patterns learned from a large amount of unlabeled data to enhance models' generalization performance. However, there are few universal patterns across different traffic datasets, *e.g.* traffic monitors in different datasets have different characteristics. So how to incorporate self-supervised learning except through the pre-training paradigm needs consideration. In addition, the self-learners in NLP are designed to capture the sequential (temporal) patterns among words while those in CV are designed to capture the spatial patterns among pixels. Compared to them, traffic data simultaneously exhibit spatial and temporal patterns. Thus how to employ self-supervised learners to enhance spatial-temporal representation learning for traffic data is another important issue.

Based on the above analysis, to address the challenges of algorithm's effectiveness and efficiency, as well as data utilization efficiency, for long-term traffic forecasting, we propose a novel spatial-temporal prediction model, called ***S**elf-supervised **S**patial-**T**emporal **B**ottleneck **A**ttentive **N**etwork* (SSTBAN). Different from existing works that only pay attention to improving the accuracy of prediction models, our work simultaneously considers the efficiency and effectiveness (prediction accuracy) of the models as well as the data utilization efficiency from an engineering point of view, so as to make it possible to deploy the complex STGNNs in real scenarios. For this goal, we design a novel building block called spatial-temporal bottleneck attention to capture the long-term spatial-temporal dynamics in a more memory-efficient and faster manner. In addition, we incorporate a specifically designed spatial-temporal self-supervised learner via the multi-task learning framework to improve the model's data utilization efficiency, bringing benefits to both the generalization ability and robustness of the prediction model. In summary, the main contributions of this work are summarized as follows:

- For the first time, we propose a novel spatial-temporal traffic forecasting model by employing a specifically designed self-supervised learner to improve data utilization efficiency, so that the model has satisfactory generalization ability and robustness.
- A novel spatial-temporal bottleneck attention mechanism is designed to effectively capture the long-term temporal and spatial dynamics in an efficient manner, which reduces the computation complexity of traditional self-attention from quadratic to linear in the lengths of time slices and the number of spatial nodes.
- We conduct nine sets of long-term traffic forecasting experiments, including three traffic speed forecasting experiments and six traffic flow forecasting experiments, on three public real-world datasets. The results show that our proposed SSTBAN enjoys advantages over the state-of-the-art baselines in both accuracy and efficiency.

II. Related Work

A. Spatial-Temporal Traffic Data Forecasting

Spatial-temporal traffic data forecasting as a key component of modern intelligent transportation systems is always a hot

topic in the field of data mining and data engineering [11], [12]. In a broad sense, traffic data as a kind of spatial-temporal data belongs to correlated time series, so classic time series analysis methods, such as AutoRegressive Integrated Moving Average [13], Vector AutoRegressive [14] are intuitive choices for traffic forecasting. However, they cannot fit well the complicated real-world traffic data with both temporal and spatial dynamics. With the development of deep learning, a series of Spatial-Temporal Graph Neural Networks (STGNNs) are proposed. Existing STGNNs are mostly tested on the short-term traffic prediction tasks with the prediction time spans shorter than an hour [7], [15], [16]. When they are applied to long-term traffic prediction, both their prediction accuracy and model efficiency face challenges. Specifically, DCRNN [17], AGCRN [18] are two representative RNN-based STGNNs. They have difficulties in capturing long-term correlations since they are based on Gated Recurrent Units (GRU) in the temporal dimension. STGCN [19], Graph Wavenet [20], ASTGCN [21], STSGCN [22], DMSTGCN [23] are well-known CNN-based STGNNs. Their ability for capturing long-term correlations are limited by the localized kernel and they have to stack multiple layers to enlarge the receptive field. GMAN [24], ASTGNN [6], ST-WA [25] are representative attention-based model, they directly capture the global temporal correlation between any two time slices via attention mechanism, meanwhile adaptively adjust the correlation strength in a flexible way, so they show advantages in most short-term traffic forecasting scenarios. However, the most adopted self-attention has quadratic computation complexity $O(n^2)$, making it impossible to deploy attention-based STGNNs in practice for long-term prediction on large traffic networks.

Until now, almost all deep-learning based models for spatial-temporal traffic data forecasting pay much attention on how to improve the prediction accuracy, *i.e.*, the model effectiveness, while few works care much about the algorithm (model) efficiency and data utilization efficiency. However, the later one is the most important problem to be solved when the models are deployed in ITS, since memory, computing resources are limited and costly. Thus, in this paper, we propose an effective and efficient deep-learning based model, which is possible to be deployed in practice from an engineering point of view.

Besides, we want to point out that this paper focuses on improving long-term traffic prediction accuracy by capturing long-term spatial-temporal traffic dynamics only relying on the recent history without attending to the periodicity. On the other hand, many works [6], [21], [26] have shown it is helpful for long-term traffic prediction by feeding the historical periodicity traffic data. However, this is not our focus. Both of the two technical routes can be utilized together to improve prediction accuracy but are conceptually different.

B. Self-Supervised Learning

Recently, self-supervised learning [8] attracts much attention for its remarkable success in representation learning in CV and NLP. It leverages input data itself as supervision without relying on manual labels, so that it largely increases data efficiency and is able to produce generalizable representations beneficial to downstream tasks.

Auto-encoder (AE) is a representative generative self-supervised framework. AE adopts an encoder to map an input to a latent representation and a decoder to reconstruct the input. Denoising autoencoders (DAE) is a class of AE that learns to reconstruct the original input from corrupted input, so that it can produce more robust representations. Masked autoencoders can be thought of generalized DAEs by masking out a portion of the input and learn to predict the removed context, which draw increasing attention recently. For example, the masked language model BERT [9] is a most representative work. It largely improves the generalization ability of NLP models. The recently proposed MAE [10] follows the high-level idea of masked auto-encoding meanwhile carefully designing the masking strategy, the encoder and the decoder according to the properties of images. Following this work, researchers study a conceptually simple extension of MAE to spatial-temporal representation learning from videos [27]. To further exploit the capability for self-supervised representation learning, CAE [28] explicitly assigns the tasks of understanding context and making predictions to its encoder and decoder respectively, proving that its design can effectively learn high-level representations for both the context and the targets in images. All these works prove that masked autoencoders are scalable self-supervised learners, which can learn robust representations that generalize well and benefit the downstream tasks.

Motivated by this, we study extending self-supervised learners to the research on spatial-temporal data forecasting. Specifically, we carefully design a spatial-temporal self-supervised learner to enhance the spatial-temporal representation learning ability for the prediction models, so as to improve the data utilization efficiency and enable the prediction models to have good generalization ability.

III. Preliminaries

We first present the definitions related to the traffic forecasting problem. Then, the frequently used notations in this paper are summarized in Table I for readers to quickly look up.

Traffic Signal Matrix. The observations on the traffic network G at time slice t are denoted by a traffic signal matrix $\mathbf{X}_t = (\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, \ldots, \mathbf{x}_{t,N}) \in \mathbb{R}^{N \times C}$, where the traffic network G consists of a set of monitor stations, denoted as a node set V and $|V| = N$. $\mathbf{x}_{t,v} \in \mathbb{R}^C$ denotes the feature vector of node v observed at time t, and C is the number of features.

Problem Statement. *Traffic Forecasting.* Given a sequence of recent historical traffic signal matrices $\boldsymbol{\mathcal{X}} = (\mathbf{X}_{t-P+1}, \mathbf{X}_{t-P+2}, \ldots, \mathbf{X}_t) \in \mathbb{R}^{P \times N \times C}$ over the past P time slices, we aim to predict the sequence of future traffic signal matrices $\boldsymbol{\mathcal{Y}} = (\mathbf{X}_{t+1}, \mathbf{X}_{t+2}, \ldots, \mathbf{X}_{t+Q}) \in \mathbb{R}^{Q \times N \times C}$ over the next Q time slices. This paper focuses on the long-term traffic forecasting, so the input time span and the output time span are longer than one hour.

TABLE I: Frequently used notations. Uppercase bold symbols denote 2D matrices. Calligraphic symbols denote 3D tensors.

Notations	Definitions
N	Number of monitor stations
C	Number of observed features
P	Length of input traffic signal matrix series
Q	Length of output traffic signal matrix series
L	Number of STBA blocks in ST Encoder
L'	Number of STBA blocks in STF Decoder
L''	Number of STBA blocks in ST Reconstructing Decoder
λ	Weight of the self-supervised learning branch
$\mathbf{X}_t$	Traffic signal matrix at time slice t
$\boldsymbol{\mathcal{X}}$	Input traffic signal matrix series
$\boldsymbol{\mathcal{Y}}$	Prediction target, *i.e.*, future traffic signal matrix series
$\boldsymbol{\mathcal{M}}$	Mask tensor
$\widetilde{\boldsymbol{\mathcal{X}}}$	Corrupted input traffic signal matrix series
$\mathbf{E}_{\mathrm{SP}}$	Spatial embedding matrix for nodes
$\mathbf{E}_{\mathrm{TP}}$	Temporal embedding matrix for input time slices
$\mathbf{E}'_{\mathrm{TP}}$	Temporal embedding matrix for output time slices
$\boldsymbol{\mathcal{E}}$	ST embedding matrix for input traffic signal matrix series
$\boldsymbol{\mathcal{E}}'$	ST embedding matrix for output traffic signal matrix series
$\boldsymbol{\mathcal{Z}}$	Input of the STBA block in ST Encoder
$\boldsymbol{\mathcal{Z}}_{:,v}$	Input about node v over all the time slices in $\boldsymbol{\mathcal{Z}}$
$\boldsymbol{\mathcal{Z}}_{t,:}$	Input about all the nodes at time slices t in $\boldsymbol{\mathcal{Z}}$
IT	Temporal reference points in the TBA of STBA block
IS	Spatial reference points in the SBA of STBA block
$\boldsymbol{\mathcal{T}}$	Output of the TBA in STBA block
$\boldsymbol{\mathcal{S}}$	Output of the SBA in STBA block
$\boldsymbol{\mathcal{H}}$	Output of the STBA block in ST Encoder with input $\boldsymbol{\mathcal{X}}$
$\boldsymbol{\mathcal{H}}'$	Output of the STBA block in STF Decoder
$\widetilde{\boldsymbol{\mathcal{H}}}$	Output of the STBA block in ST Encoder with input $\widetilde{\boldsymbol{\mathcal{X}}}$
$\widetilde{\boldsymbol{\mathcal{H}}'}$	Output of the STBA block in ST Reconstructing Decoder

Multi-Head Self-Attention. It is a fundamental operation in our model. which first projects the queries, keys and values into h different d-dimensional subspaces and then performs the attention function in parallel. Finally, the outputs are concatenated and further projected. Formally,

$$\mathrm{MHSA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \oplus(\mathrm{head}_1,\ldots,\mathrm{head}_h)W^O,$$
$$\mathrm{head}_j = \mathrm{softmax}(\frac{(\mathbf{Q}W_j^Q)(\mathbf{K}W_j^K)^\top}{\sqrt{d}})(\mathbf{V}W_j^V).$$

IV. Self-Supervised Spatial-Temporal Bottleneck Attentive Network

The high-level idea of our proposed model is to employ a self-supervised learner to exploit the capability of the encoder for making full use of limited training data and improving the qualities of its output representations. As shown in Figure 1, we incorporate a spatial-temporal self-supervised learner into the prediction model via a multi-task learning framework. Specifically, our model have two branches, one branch is designed for the spatial-temporal forecasting task, and the other branch is designed for the spatial-temporal self-supervised learning task. These two branches share the same spatial-temporal encoder, whose role is to generate the latent representation of its input by understanding the global spatial-temporal context. In addition, to effectively capture the long-term spatial-temporal dynamics of traffic data at low computation cost, we propose a novel kind of attention, named spatial-temporal bottleneck attention, as the building block of each branch.

A. Training and Testing Pipeline

The pipeline of SSTBAN works as follows.

During the training stage, two branches work together. In the first spatial-temporal forecasting branch, the original unbroken input $\boldsymbol{\mathcal{X}}$ is fed into a spatial-temporal (ST) encoder, a transformer attention and a ST forecasting decoder in succession. The role of ST encoder is to understand the spatial-temporal dynamics from historical data and the role of ST forecasting decoder is to make predictions in future.

At the same time, we randomly mask out some patches in the original input to get a corrupted input $\widetilde{\boldsymbol{\mathcal{X}}}$ as the input to the second spatial-temporal self-supervised learning branch. In this branch, the corrupted input is fed into a ST encoder and a ST reconstructing decoder in succession. This branch is actually a MAE learner, the ST encoder is responsible for understanding global spatial-temporal patterns by only using the remaining patches, and the ST reconstructing decoder is responsible for generating a plausible guess for the original unbroken input. In addition, we let this guess is performed on the latent space, so we align the output of the ST reconstructing decoder with the output of ST encoder in the first branch. Alignment in the latent space avoid the noise from the observations, helping the model to focus on capturing the high-level spatial-temporal patterns of the traffic data.

The training loss consists of two parts, in which one part adopts mean absolute error to measure the prediction accuracy of the spatial-temporal forecasting branch, and the other one is the alignment loss using mean square error to measure the distance between two latent representations. The two parts are summed up with weights $(1-\lambda)$ and λ.

In the two branches, the encoder and the two decoders consists of identical spatial-temporal bottleneck attentive blocks (STBA) accompanied with spatial-temporal embedding (STE) blocks.

The STBA block aims at simultaneously capturing the long-term spatial and temporal dynamics through an efficient self-attention, meanwhile keeping the computational complexity at a low level. We will detailedly illustrate it in Section IV-B.

The STE block enables the model to be aware of the unique characteristics of different time slices and spatial nodes. Thus, introducing the STE block makes up for the drawback of attention-based STBA blocks that they are completely agnostic to the order information. For the spatial embedding $\mathbf{E}_{\mathrm{SP}} \in \mathbb{R}^{N\times d}$, we learn a vector for each node in the traffic network by end-to-end training and it is shared among time slices. For the temporal embedding, we learn a vector for each time slice by inputting the time-of-day and the day-of-week feature through one-hot encoding and a MLPs. We use $\mathbf{E}_{\mathrm{TP}} \in \mathbb{R}^{P\times d}$ to denote the learned temporal embedding of

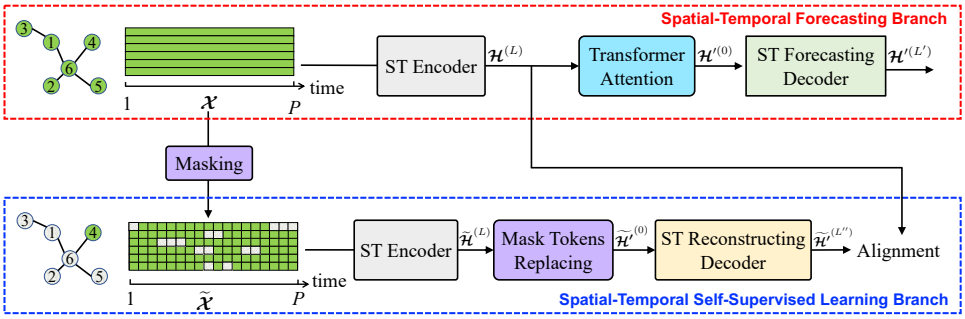


Fig. 1: The architecture of SSTBAN.

input sequence and use $\mathbf{E}'_{\mathrm{TP}} \in \mathbb{R}^{Q\times d}$ to denote the learned temporal embedding of output sequence. $\mathbf{E}_{\mathrm{TP}}$ and $\mathbf{E}'_{\mathrm{TP}}$ are shared among nodes. We finally term the sum of spatial and temporal embedding as $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{P\times N\times d}$ for the input sequence and $\boldsymbol{\mathcal{E}}' \in \mathbb{R}^{Q\times N\times d}$ for the output sequence.

B. Spatial-Temporal Bottleneck Attention

Self-attention is an effective way to model the long-term spatial-temporal dynamics [6], [24]. However, the quadratic computation complexity of self-attention caused by the direct connections between any two elements, hinders its application in practice, especially when the traffic networks have more than a hundred sensors or the traffic series covering more than ten time slices. In order to bypass the efficiency problem, we propose a novel spatial-temporal bottleneck attention, inspired by [29].

As shown in Figure 2, the STBA block contains a spatial bottleneck attention (SBA) and a temporal bottleneck attention (TBA). Instead of directly connecting every two elements in the traffic series, we introduce spatial and temporal reference points to bridge the connections between elements respectively along the spatial and temporal dimensions. More importantly, we expect the introduced reference points could encode the generalized global patterns of traffic data. Since the number of reference points is much smaller than that of original elements and this structure looks like a bottleneck, we call it spatial-temporal bottleneck attention. The forward computations are as follows.

Denote the output of $(l-1)^{\mathrm{th}}$ STBA block as $\boldsymbol{\mathcal{H}}^{(l-1)} \in \mathbb{R}^{P\times N\times d}$. We concatenate it with the spatial-temporal embedding, *i.e.*, $\boldsymbol{\mathcal{Z}}^{(l-1)} = (\boldsymbol{\mathcal{H}}^{(l-1)}||\boldsymbol{\mathcal{E}}) \in \mathbb{R}^{P\times N\times 2d}$, and feed $\boldsymbol{\mathcal{Z}}^{(l-1)}$ into the l^{th} STBA block. Besides, we use $\boldsymbol{\mathcal{Z}}^{(l-1)}_{:,v} \in \mathbb{R}^{P\times 2d}$ to denote the input about node v over all the time slices, and $\boldsymbol{\mathcal{Z}}^{(l-1)}_{t,:} \in \mathbb{R}^{N\times 2d}$ to denote the input about all the nodes at time slice t.

In the temporal dimension, we additionally define T' vectors $\mathbf{IT}^{(l)} \in \mathbb{R}^{T'\times 2d}$ as temporal reference points in the TBA of l^{th} STBA block, which are all trainable parameters of the model. Then the computations in TBA are as follows,

$$\begin{aligned}\mathbf{IT}' &= \mathrm{MHSA}(\mathbf{IT}^{(l)}, \boldsymbol{\mathcal{Z}}^{(l-1)}_{:,v}, \boldsymbol{\mathcal{Z}}^{(l-1)}_{:,v}) \in \mathbb{R}^{T'\times 2d},\\ \mathbf{T}^{(l)}_{:,v} &= \mathrm{MHSA}(\boldsymbol{\mathcal{Z}}^{(l-1)}_{:,v}, \mathbf{IT}', \mathbf{IT}') \in \mathbb{R}^{P\times d}.\end{aligned} \tag{1}$$

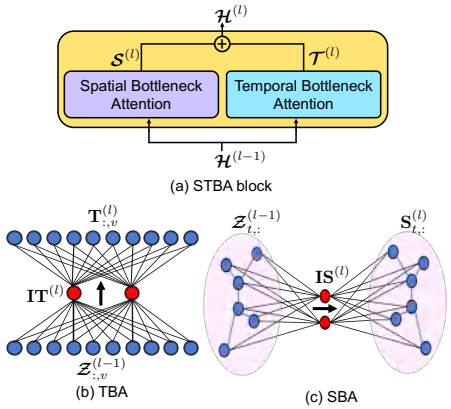


Fig. 2: The illustration of (a) spatial-temporal bottleneck attention, (b) temporal bottleneck attention, and (c) spatial bottleneck attention. The red points in (b), (c) are the introduced learnable reference points.

The TBA first transforms $\mathbf{IT}^{(l)}$ into $\mathbf{IT}'$ by attending to the input $\boldsymbol{\mathcal{Z}}^{(l-1)}_{:,v}$. The updated temporal reference points $\mathbf{IT}'$, which contains global information about the original long-term input $\boldsymbol{\mathcal{Z}}^{(l-1)}_{:,v}$, is again attended to by the original input $\boldsymbol{\mathcal{Z}}^{(l-1)}_{:,v}$ to finally produce the temporal encoding representation $\mathbf{T}^{(l)}_{:,v}$ of node v. The TBA block processes the input about each node in parallel. We use $\boldsymbol{\mathcal{T}}^{(l)} \in \mathbb{R}^{P\times N\times 2d}$ to denote the temporal encoding representation of all the nodes.

Likewise, in the spatial dimension, N' vectors $\mathbf{IS}^{(l)} \in \mathbb{R}^{N'\times 2d}$ are introduced as the spatial reference points, then the SBA processes the input at each time slice in parallel,

$$\begin{aligned}\mathbf{IS}' &= \mathrm{MHSA}(\mathbf{IS}^{(l)}, \boldsymbol{\mathcal{Z}}^{(l-1)}_{t,:}, \boldsymbol{\mathcal{Z}}^{(l-1)}_{t,:}) \in \mathbb{R}^{N'\times 2d},\\ \mathbf{S}^{(l)}_{t,:} &= \mathrm{MHSA}(\boldsymbol{\mathcal{Z}}^{(l-1)}_{t,:}, \mathbf{IS}', \mathbf{IS}') \in \mathbb{R}^{N\times d}.\end{aligned} \tag{2}$$

where $\mathbf{S}^{(l)}_{t,:}$ is the spatial encoding representation of all the nodes at time t. In addition, $\boldsymbol{\mathcal{S}}^{(l)} \in \mathbb{R}^{P\times N\times 2d}$ denotes the spatial encoding representations over all the time slices. Finally, the output of the STBA block $\boldsymbol{\mathcal{H}}^{(l)}$ is the sum of $\boldsymbol{\mathcal{S}}^{(l)}$ and $\boldsymbol{\mathcal{T}}^{(l)}$.

The STBA block has two strong points. We take SBA for example to illustrate these in detail. Firstly, the SBA enjoys global attention connections between every two nodes

with computational complexity $O(NN')$. Since N' is a typically small hyper-parameter, this is an improvement over the quadratic complexity $O(N^2)$ of traditional self-attention. Compare to other aggregation operations like ChebNet [30], GCN [31], our bottleneck attention in the spatial dimension does not rely on the pre-defined graph structure, meanwhile dynamically adjusting the relation strength between nodes. Secondly, the bottleneck attention architecture is analogous to low-rank projection, where the inputs are first projected onto a low-dimensional object and then reconstructed to produce outputs. The learned reference points are expected to encode some global patterns which helps explain the input sequence. *i.e.*, the spatial reference points can be regarded as cluster centers. Hence, the SBA compares every two nodes in the original input indirectly through their proximity to these cluster centers.

C. Spatial-Temporal Forecasting Branch

The goal of the spatial-temporal forecasting branch is to make predictions in future. As shown in Figure 3, it consists of an ST encoder, a transformer attention block and an ST forecasting decoder. Next, we introduce them in detail.

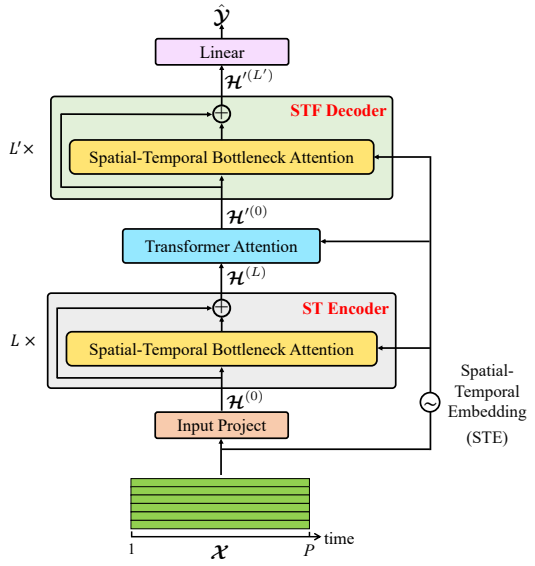


Fig. 3: The illustration of spatial-temporal forecasting branch.

1) Spatial-Temporal Encoder: The raw input $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{P\times N\times C}$ to the spatial-temporal forecasting branch is first projected into $\boldsymbol{\mathcal{H}}^{(0)} \in \mathbb{R}^{P\times N\times d}$ via a linear transformation. Then $\boldsymbol{\mathcal{H}}^{(0)}$ is fed into the ST encoder, which stacks L STBA blocks with residual connections. The ST encoder is designed to embed the long-term spatial-temporal dynamics of input traffic data into a latent representation $\boldsymbol{\mathcal{H}}^{(L)}$.

2) Transform Attention: We introduce a transform attention block to convert the latent representations of historical traffic signals to meet the temporal dimension size of predicted future signals. Specifically, to alleviate the error propagation issue, which is extremely outstanding in long-term prediction, we follow [24] to employ an attention mechanism to directly connect the historical traffic signal at each step with the predicted traffic signal at each step by adaptively fusing different features over the history. In addition, the transform attention makes it more flexible to predict future traffic with varying time spans. The transformation is performed in parallel among nodes. Formally,

$$\boldsymbol{\mathcal{H}}'^{(0)}_{:,v} = \mathrm{MHSA}(\boldsymbol{\mathcal{E}}'_{:,v}, \boldsymbol{\mathcal{E}}_{:,v}, \boldsymbol{\mathcal{H}}^{(L)}_{:,v}) \in \mathbb{R}^{Q\times d}, \tag{3}$$

where $\boldsymbol{\mathcal{E}}'_{:,v} \in \mathbb{R}^{Q\times d}$ and $\boldsymbol{\mathcal{E}}_{:,v}, \boldsymbol{\mathcal{H}}^{(L)}_{:,v} \in \mathbb{R}^{P\times d}$. That is to say, the correlation strengths between the history and the future are measured by the spatial-temporal embedding.

3) Spatial-Temporal Forecasting Decoder: The input to ST forecasting decoder is the output of transform attention, *i.e.*, $\boldsymbol{\mathcal{H}}'^{(0)}$. The ST forecasting decoder first stacks L' STBA to further capture the spatial-temporal dynamics in the latent space, then a linear projection is applied to map the latent representations to the target future traffic signals.

D. Spatial-Temporal Self-Supervised Learning Branch

The spatial-temporal self-supervised learning branch is essentially a MAE learner. It aims to reconstruct the original complete signals from corrupted inputs. Specifically, some patches of the original complete traffic signals are firstly masked out. Then, the ST encoder in this branch tries to understand the spatial-temporal context of input only relying on the remaining signals, and maps the observed signal to a latent representation. Finally, the decoder in this branch reconstructs the original signal by using the latent representation and the introduced mask tokens. Next we will detail the masking strategy and the implementation of encoder and decoder.

1) Masking: The masking strategy for spatial-temporal traffic data is different from those in CV/NLP. Firstly, the information density of spatial-temporal traffic data sequences is much lower than that of language in NLP. The words in sentences are highly semantic and information-dense, so masking discrete words per sentence encourages the model to capture the semantic meaning of the whole sophisticated sentence. On the contrary, traffic data has redundancy along the temporal dimension, *i.e.*, a discrete missing value can be easily recovered from neighboring time windows without comprehensive modeling of the changing trend of the traffic time series. Considering this difference, to encourage learning useful changing trend patterns, we should mask out a continuous segment in traffic sequence to build the self-supervised learning task. Secondly, the masking should be made along both the temporal and spatial dimensions to capture the spatial-temporal dependencies. By contrast, languages only show temporal patterns in NLP. Grid-based images only show spatial patterns in CV.

Thus, we propose a masking algorithm for spatial-temporal traffic data as shown in Algorithm 1. We first split the traffic signal matrices into patches, *i.e.*, a continuous segment of signals with length l_m. Then we randomly mask out patches with spacetime-agnostic sampling. Compared to the spacetime-agnostic sampling strategy, we could also sample

Algorithm 1 Mask Tensor Generation.

Input: the historical traffic signal matrices $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{P \times N \times C}$;
Parameter: patch length l_m, masking rate α_m;
Output: mask tensor $\boldsymbol{\mathcal{M}}$;

```
generate a mask tensor M ∈ R^{P×N×C} and set all its elements to 1;
split signal matrices into patches and index them with {1, 2, ..., N_patch}, where N_patch = PNC / l_m;
calculate masked patches number n_m = ⌊α_m N_patch⌋;
randomly sample n_m indices without replacement from {1, 2, ..., N_patch}, and the corresponding patch index set is denoted as P_sample = {p_1, p_2, ..., p_{n_m}}
for p_i in P_sample do
    set all the elements of p_i-th patches in M to zero;
end for
return mask tensor M;
```

patches only along the spatial or temporal dimension. And we conduct experiments in Section V-D4 to find out which mask sampling strategy is better.

Finally, we get $\widetilde{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{X}} \odot \boldsymbol{\mathcal{M}}$ as the input to the spatial-temporal self-supervised learning branch, where $\odot$ means element-wise product.

2) Spatial-Temporal Encoder: The encoder in the spatial-temporal self-supervised learning branch is just the same one used in the spatial-temporal forecasting branch, since we expect the self-supervised learning task could enhance the learning ability of the ST encoder. In this ST encoder, the masked out traffic signals are not involved in the computation by setting the masked values to $-\infty$ in the input of the softmax function. The output of this ST encoder is denoted as $\widetilde{\boldsymbol{\mathcal{H}}}^{(L)}$.

3) Spatial-Temporal Reconstructing Decoder: The input to the reconstructing decoder is noted as $\widetilde{\boldsymbol{\mathcal{H}}'}^{(0)}$, which consists of two parts. One part is the latent representation outputted from the above ST encoder. The other part is mask token vectors occupying all the masked out positions to reveal the absence of true values. The mask token vectors are shared and trained along with the whole model, indicating the presence of masked patches to be recovered.

Reconstructing decoder stacks L'' STBA blocks to recover the traffic signals in the latent space, and finally outputs $\widetilde{\boldsymbol{\mathcal{H}}'}^{(L'')}$. The reconstructed latent representations for the original complete input are constrained to match with the latent representations $\boldsymbol{\mathcal{H}}^{(L)}$ computed from the ST encoder in the spatial-temporal forecasting branch.

There are two reasons for making the alignment in the latent space. Firstly, the traffic signals may have some noises, so making alignment in the original data level will make the model vulnerable to noise. Secondly, such alignment encourages that the latent representation coded by the ST encoder could infer the global spatial-temporal dynamics of both the masked out signals and the remaining signals.

V. Experiments

To evaluate the performance of our proposed model, we conduct nine sets of experiments on three real-world spatial-temporal traffic graph datasets.

A. Datasets

We evaluate SSTBAN on two kinds of traffic forecasting tasks, including traffic speed forecasting and traffic flow forecasting, to make a comprehensive understanding of our model.

(1) The first kind of task is traffic speed prediction on the **Seattle Loop** dataset [32]. This dataset contains 1 year of traffic data recorded by 323 loop detectors ranging from January 1st, 2015 to December 31st, 2015 on freeways in Seattle area. The observations contain three features, *i.e.*, $C = 3$, including flow, speed and occupancy. We aggregate data into 1-hour intervals by summing up the flow and averaging the speed and occupancy.

(2) The second kind of task is traffic flow prediction. We choose two widely used datasets, including **PEMS04** and **PEMS08**. The datasets are collected by the Caltrans Performance Measurement System (PeMS) [33] and are aggregated every 5 minutes. Specifically, PeMSD4 has 307 detectors and PeMSD8 has 170 detectors. Following [6], [22], only traffic flow is used, *i.e.*, $C = 1$.

The detailed statistical information of datasets is summarized in Table II.

TABLE II: Dataset Description.

Task type	Datasets	# of nodes	Time range	Time interval
speed forecasting	Seattle	323	01/01/2015 - 12/31/2015	1 hour
flow forecasting	PEMS04	307	01/01/2018 - 02/28/2018	5 minutes
	PEMS08	170	07/01/2016 - 08/31/2016	5 minutes

B. Baseline Methods

We compare SSTBAN with the following baseline methods:
(1) **HA**: directly uses the average of inputted historical data as the prediction.
(2) **VAR** [34]: Vector Auto-Regression is an advanced time series model, which captures the pairwise relationships among multiple time series.
(3) **DCRNN** [17]: combines diffusion convolutions with GRU to make predictions with encoder-decoder.
(4) **GWNet** [20]: combines graph convolutions with dilated casual convolutions to capture spatial-temporal dependencies.
(5) **GMAN** [24]: employs multi-attention in both spatial and temporal dimensions to capture spatial-temporal dynamics, along with spatial and temporal embeddings to incorporate graph structure and time information.
(6) **AGCRN** [18]: captures node-specific patterns by a node adaptive parameter learning module, and designs a adaptive graph convolutional recurrent network to capture spatial-temporal dependencies.
(7) **ASTGNN** [6]: designs trend-aware self-attention module and a dynamic graph convolution module to capture spatial-temporal dynamics.

(8) **DMSTGCN** [23]: proposes a spatial dependencies learning method to dynamically adjust the correlations between nodes, along with a multi-faceted fusion module to deal with multi features of traffic data.

HA is the most simple and straightforward method. DCRNN is the most representative and early proposed method. GWNet, GMAN is two widely adopted and competitive methods. AGCRN, ASTGNN and DMSTCGN are the most recently proposed methods. Most of them are only tested on one forecasting task (speed or flow) within 1 hour. Compared to them, we test on both the two tasks, and extend the prediction to more than 1 hour, which is more useful in practice.

C. Settings

Our experiments are conducted on a server with Hygon C86 7151 16-core Processor and NVIDIA RTX A4000 GPU cards. All the STGNNs are implemented by PyTorch[1]. The data and code for SSTBAN has been released on GitHub[2]. We split all datasets with ratio $6:2:2$ into training sets, validation sets, and test sets by the time. We pre-process data by standard normalization. For evaluation, we re-transform the predictions back to the actual values and compare them with the ground truth. Mean absolute error (MAE), root mean square error (RMSE) and mean absolute percentage error (MAPE) are selected as the evaluation metrics. To avoid over-fitting, we adopt early-stopping strategy with patience at 5.

As this paper focuses on long-term traffic forecasting, and aims to compare the ability of STGNNs to capture the long-term spatial-temporal dynamics on long traffic sequences, we set three prediction scenarios, *i.e.*, $P = Q = 24$, $P = Q = 36$, $P = Q = 48$. So on the **Seattle Loop**, we make predictions for next 1, 1.5, 2 days; on the **PEMS04 and PEMS08**, we make predictions for next 2, 3, 4 hours.

We choose the hyper-parameters for the models by the performance on the validation sets. For all the nine scenarios, we set T', N' the number of temporal and spatial reference points in STBA blocks to 3, 3 for our SSTBAN. As proven in [10], a narrow decoder is enough for the MAE task, so we set L'' to 1. Batch size is 4. Learning rate is 0.001. Other detailed settings for each scenarios are described in Table III.

TABLE III: SSTBAN settings.

Scenario	Encoder/Decoder				Self-supervised Task		
	L	L'	d	h	l_m	α_m	λ
Seattle-24	4	4	4	8	3	0.3	0.1
Seattle-36	2	2	8	16	18	0.5	0.5
Seattle-48	2	2	8	16	3	0.3	0.1
PEMS04-24	2	2	16	8	12	0.1	0.05
PEMS04-36	2	2	16	8	12	0.3	0.05
PEMS04-48	2	2	16	8	3	0.2	0.3
PEMS08-24	3	3	16	8	12	0.1	0.05
PEMS08-36	3	3	16	8	12	0.5	0.8
PEMS08-48	3	3	16	8	24	0.5	0.3

[1]https://pytorch.org

[2]https://github.com/guoshnBJTU/SSTBAN

D. Results Analysis

1) Overall Comparison: Table IV and Table V present the average results on the two tasks over the next 24, 36 and 48 steps. Since the errors of traffic speed forecasting are relatively small, so we report both the mean and standard deviations. From the results, we observe that:

(1) SSTBAN almost achieves the best performances on Seattle Loop under all the three scenarios, except the RMSE on 24 steps prediction. Specifically, HA that only takes the average of history as the prediction results is the worst choice, denoting that designing methods to effectively capture the complicated patterns of spatial-temporal traffic data is necessary. The prediction accuracy of VAR, a kind of time series analysis model, is significantly lower than that of deepl-learning based STGNNs, proving that the spatial-temporal dynamics of traffic data is so complicated that cannot be pre-defined. Deep-learning based STGNNs are always the better choice for long-term traffic forecasting.

Among STGNNs, GMAN and DMSTGCN is the best baselines on Seattle Loop, and DMSTGCN shows better results on predictions with shorter time spans, while GMAN shows better results on the predictions with longer time spans. The good performance of GMAN on 36 steps and 48 steps predictions proves the effectiveness of attention mechanism on capturing the long-term dynamics, despite of the high computational costs. Compared to GMAN, our SSTBAN further improves prediction accuracy on 36 steps prediction (resp. 48 steps prediction) by 2.88%, 1.83%, 3.94% (resp. 3.38%, 2.62%, 7.60%) in terms of MAE, RMSE, MAPE.

The data frequency of Seattle Loop is 1 hour, so all the 24, 36, 48 steps predictions cover several continuous day. So these results prove the superiority of SSTBAN on modeling the long-term dynamics. *i.e.*, SSTBAN can effectively learn the daily patterns from the inputs. The left column in Figure 4 shows the changing of prediction errors on the 36-step prediction task on the Seattle Loop dataset. We find that the advantages of our SSTNBAN becomes obvious as the prediction span extend.

(2) SSTBAN is the most competitive model under most scenarios on the PEMS04. Similar to the results on Seattle Loop, DMSTGCN and GMAN is the most competitive model. Taking the results on 36-steps prediction for example, compare to the second best baseline, our SSTBAN further decrease the errors by 7.34%, 2.06%, 6.48% in terms of MAE, RMSE and MAPE. From the prediction errors shown in Table IV and Table V, we find that the errors of traffic flow prediction is much larger than those of traffic speed prediction. This phenomenon is in line with the actual situation on the highway. Since highway usually has specific speed limitations, traffic speed usually does not change dramatically and the variations of speed are not large. By contrast, traffic flow is more complicated and its fluctuations are more volatile, so the traffic flow predictions is more difficult and the errors are larger. Despite that, our SSTBAN is always the best, no matter what kind of task is.

TABLE IV: Performance comparison on traffic speed forecasting under three scenarios.

Dataset	Scenario	24 steps			36 steps			48 steps		
	Method	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
Seattle Loop	HA	8.08	11.86	26.54	8.50	12.35	27.68	8.53	12.30	27.76
	VAR	6.22	9.33	18.58	6.29	9.57	19.54	6.45	9.87	20.49
	DCRNN	4.37 ± 0.04	7.97 ± 0.08	14.04 ± 0.23	4.60 ± 0.04	8.38 ± 0.07	14.41 ± 0.36	4.73 ± 0.11	8.63 ± 0.16	14.91 ± 0.62
	GWNet	4.28 ± 0.11	7.84 ± 0.18	14.06 ± 0.58	4.60 ± 0.19	8.18 ± 0.29	15.12 ± 0.96	4.67 ± 0.14	8.35 ± 0.25	15.04 ± 0.82
	GMAN	4.13 ± 0.12	7.84 ± 0.24	12.88* ± 0.94	4.23* ± 0.08	8.10 ± 0.16	12.95* ± 0.54	4.26* ± 0.05	8.09* ± 0.11	13.26* ± 0.35
	AGCRN	4.27 ± 0.07	7.83 ± 0.11	13.53 ± 0.30	4.66 ± 0.11	8.31 ± 0.07	14.76 ± 0.14	4.82 ± 0.05	8.60 ± 0.14	15.62 ± 0.72
	DMSTGCN	4.08* ± 0.05	**7.59 ± 0.10**	13.51 ± 0.24	4.31 ± 0.03	7.98* ± 0.10	14.31 ± 0.02	4.49 ± 0.05	8.20 ± 0.06	14.86 ± 0.32
	ASTGNN	4.26 ± 0.07	8.31 ± 0.16	13.64 ± 0.52	4.78 ± 0.06	9.11 ± 0.09	15.29 ± 0.32	5.15 ± 0.12	9.58 ± 0.22	16.93 ± 0.64
	SSTBAN	**4.05 ± 0.12**	7.72* ± 0.23	**12.69 ± 0.69**	**4.11 ± 0.05**	**7.83 ± 0.11**	**12.44 ± 0.38**	**4.12 ± 0.03**	**7.88 ± 0.05**	**12.25 ± 0.33**

TABLE V: Performance comparison on traffic flow forecasting under three scenarios.

Dataset	Scenario	24 steps			36 steps			48 steps		
	Method	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
PEMS04	HA	56.47	81.57	45.49	76.01	106.58	68.84	93.37	127.28	94.62
	VAR	27.19	41.09	21.42	30.48	45.44	24.51	33.5	49.46	27.28
	DCRNN	28.70	42.86	21.23	33.78	51.40	27.10	38.26	57.85	33.73
	GWNet	22.79	35.52	16.04	24.71	38.17	17.67	26.42	40.60	18.99
	GMAN	21.67	38.10	17.78	22.12*	52.86	16.43	23.35	47.85	17.98
	AGCRN	21.63	34.44	14.65	24.15	38.19	16.33	24.18	38.26	16.31*
	DMSTGCN	20.32*	**32.09**	**14.13**	22.47	34.86*	15.86*	22.50*	**35.05**	16.56
	SSTBAN	**20.17**	32.82*	14.43*	**20.82**	**34.15**	**14.83**	**21.66**	35.51*	**15.90**
PEMS08	HA	48.3	69.72	32.09	65.99	92.72	46.64	81.51	111.85	61.29
	VAR	28.31	44.47	19.53	31.7	48.96	22.56	34.51	52.14	25.28
	DCRNN	22.60	33.34	15.46	25.82	39.37	18.53	30.47	45.64	25.10
	GWNet	19.07	29.47	12.25	21.76	33.54	13.68	22.60	34.20	14.16
	GMAN	17.38	34.29	15.66	17.21*	35.89	16.33	18.70	48.54	16.81
	AGCRN	17.45	28.05	**11.25**	19.39	30.96	12.73	19.46	31.11	12.88*
	DMSTGCN	16.75*	26.55*	11.44*	18.15	28.50*	12.64*	18.34*	28.94*	12.93
	SSTBAN	**15.97**	**26.32**	12.29	**16.84**	**28.30**	**12.20**	**16.94**	**28.82**	**12.47**

(3) On PEMS08, SSTBAN is always the best model under the three settings on prediction time span. Besides, its MAE on the 36-steps prediction and 48-steps prediction is largely lower than the second best models, improving 7.20% and 8.18% respectively. The second column in Figure 4 shows the changing of prediction errors on the 36-step prediction task on PEMS08.

2) Robustness Test: To practically deploy STGNNs for long-term traffic forecasting, we should test their robustness, as in real scenarios the high requirement from STGNNs on the quantity and quality of training data may not be met.

To test the robustness of the three best models on long-term traffic forecasting, *i.e.*, our SSTBAN, GMAN, and DMSTGCN, we first reduce the training data from 60% of the whole dataset to 40% and 20% by dropping out the data starting from the earliest time slice while keeping the test and validation datasets the same. From results shown in Figure 5, we find that our SSTBAN is always the best one no matter how the training data sizes vary.

Additionally, in real scenarios, it is common that the recorded data is disturbed owing to network transmission errors and detector errors. So we conduct experiments to figure out the best robust model when the training data is disturbed. Specifically, we randomly add noise sampled from a Gaussian distribution (with the mean at 10 and the standard deviation at 500) to 10%, 30%, and 90% input data respectively. Figure 6 shows the model's performance. We can still find that our SSTBAN is always the best model when the qualities of training data vary.

3) Effect of Network Configurations: We first conduct an ablation study on the STBA block to figure out how it affects the efficiency and effectiveness of the prediction model. Specifically, we replace the STBA in SSTBAN with traditional attention and keep other hyper-parameters the same (as shown in Table III. However, we find that such models cannot run on a NVIDIA RTX A4000 GPU card as it reports "Out of Memory" error. This phenomenon proves that our proposed STBA does largely reduce memory costs and lets it possible to practically apply the models and tune the hyper-parameters.

TABLE VI: Ablation study on the STBA block.

Scenario	model	MAE	RMSE	MAPE (%)
Seattle-36	SSTBAN	4.11	7.83	12.44%
	w/o STBA	4.16	7.91	12.84%
PEMS08-36	SSTBAN	16.84	28.30	12.20%
	w/o STBA	17.29	35.61	16.27%

Limited by the memory size, we adjust L, L' to 1 and keep other settings the same. Then the experimental results are shown in Table VI. We can find that SSTBAN with the STBA is largely superior to the degraded version without the STBA, proving the effectiveness and efficiency of the STBA.

Then we investigate the influences of hyper-parameter settings. More specifically, we study the hyper-parameter about

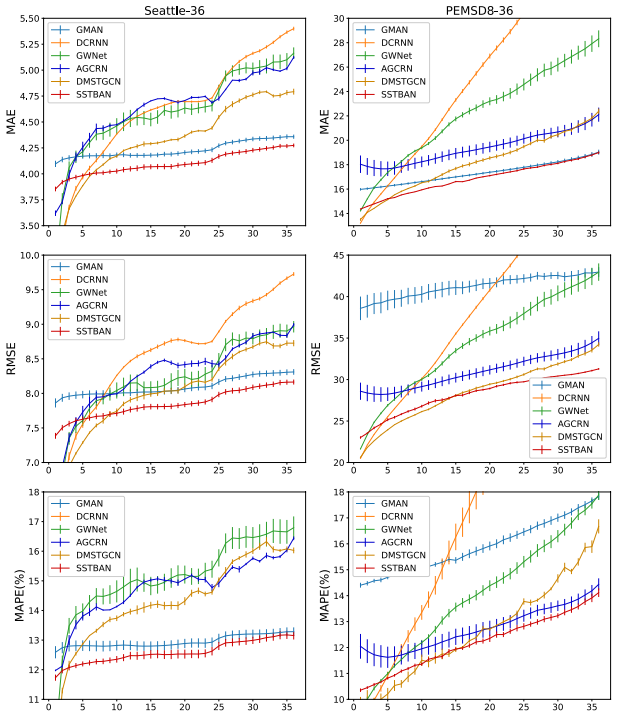


Fig. 4: Performance changes as the forecasting spans increase.

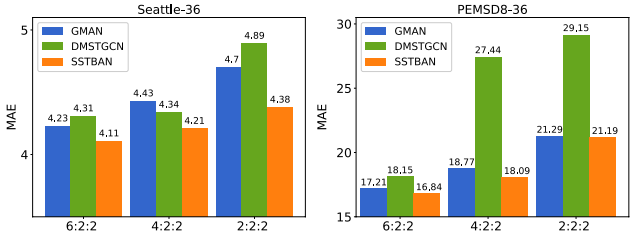


Fig. 5: Performance changes as the training data sizes decrease.

STBA, the hidden representation size d, the number of STBA blocks L, the number of attention head h, the number of spatial and temporal reference points T'/N', and those about the spatial-temporal self-supervised learner, including the weigh of self-supervised learning branch λ, patch length l_m on the Seattle Loop-36 steps forecasting task and the PEMS08-36 steps forecasting task.

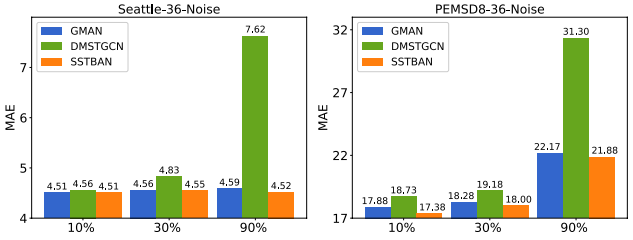


Fig. 6: Performance changes as the training data qualities decrease.

Figure 7 illustrates the changes of results as hyper-parameters changes. We can find that large d and L improve the performance on PEMS08-36. The experimental results on Seattle-36 and PEMS08-36 show that large T'/N' is not necessary since fewer reference points not only speed up the model but also improve the prediction performance (MAE) slightly.

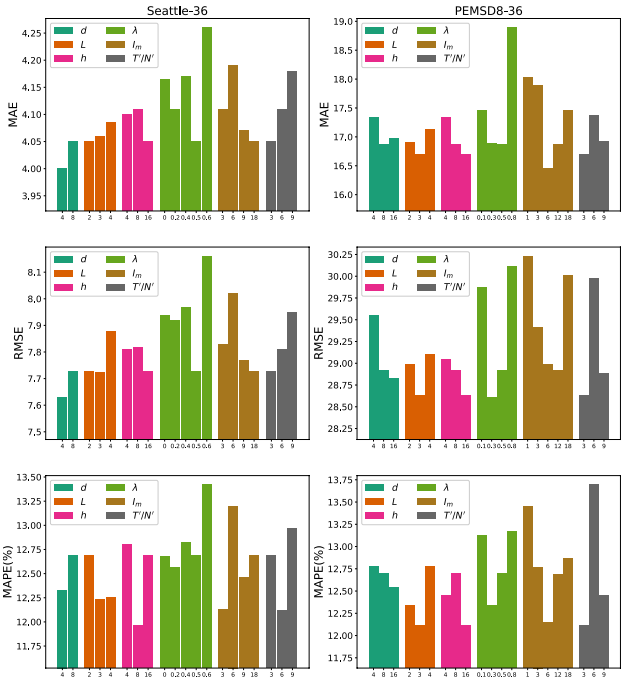
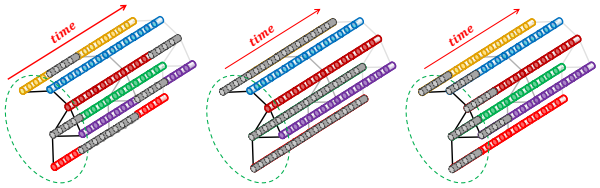


Fig. 7: Performance changes as network configurations change.

4) Mask Sampling Strategy: Mask sampling on spatial-temporal graph traffic data has three choices. The first choice is spacetime-agnostic mask, which is described in Algorithm 1. The second one is space-only random sampling, which only selects nodes to be masked out in the spatial dimension, and then is broadcasted to all time steps. The last choice is time-only random sampling, which only selects time slices to be masked out, and then is broadcasted to all the spatial nodes. Figure 8 shows the differences between three mask sampling strategies.



(a) spacetime-agnostic (b) space-only (c) time-only

Fig. 8: The illustrations of three mask sampling strategies. (The gray segment means the corresponding patches have been masked out.)

We compare the three strategies on PEMS04 and PESM08. All the hyper-parameter settings are the same as that shown in

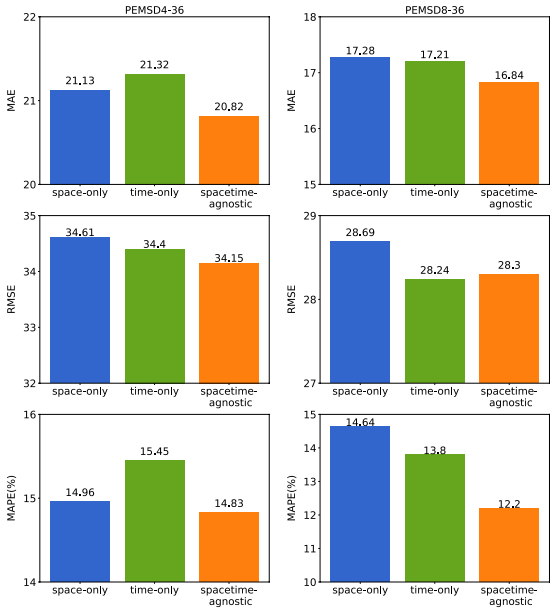


Fig. 9: Performance changes as mask sampling strategy changes.

Table III except the mask sampling strategy. The experimental results are illustrated in Figure 9. The most straightforward spacetime-agnostic mask sampling achieves the best performance. We think the reason is that both the space-only or time-only mask sampling strategies improve the difficulties of the self-supervised learning task, thus yield an unbalanced task that may harm the performance of the forecasting task.

5) Computation Cost: The computation cost is the most important factor affecting the application of STGNNs in practice. Here, we analyze the computational complexity of our proposed SSTBAN. Specifically, from a theoretical perspective, the computational complexity of temporal bottleneck attention is $O(PT')$ and the computational complexity of spatial bottleneck attention is $O(NN')$. Besides, the attention can be sped up by parallel computing. Thus, the training cost of SSTBAN grows linearly with the number of time slices and nodes in the traffic network.

From an experimental perspective, we make comprehensive comparisons on the per epoch training time, the total training time, the inference time and the GPU memory costs of SSTBAN as well as all the baselines. Table VII shows the results of the Seattle-36 scenario. We can observe that our model has the shortest running time (adding the total training time and total inference time together). The GPU cost of our SSTBAN is not the smallest but still acceptable, since our model has an additional branch for self-supervised learning. More importantly, due to the self-supervised learner, our model has the best generalization ability compared to all the baselines.

TABLE VII: Computation cost.

Methods	total inference time (s)	training time (s/epoch)	total training time (s)	**total running time (s)**	**total GPU cost (M)**
DCRNN	123	1014	14314	**14437**	**1331***
GWNet	32	289	4979	**5011**	**2597**
GMAN	77	728	8856	**8933**	**14271**
AGCRN	69	478	12458	**12527**	**7953**
DMSTGCN	50	531	15980	**16030**	**5747**
ASTGNN	197	904	21341	**21538**	**16089**
SSTBAN	42	774	4089	**4131***	**9585**

VI. Conclusion

In this paper, we propose a novel SSTBAN to address the long-term traffic forecasting problem. Aiming at deploying the algorithm in ITS in practice, one of the most important problem is to improve the efficiency and effectiveness of forecasting models. To solve this problem, we design a novel kind of spatial-temporal bottleneck attention mechanism to simultaneously enable the model to capture the long-term spatial-temporal dynamics and keeps the computational cost at a low level. The other important problem we have to face in practice is the data utilization efficiency, since data resource is constrained. Thus, to make full use of training data and to improve data utilization efficiency of the forecasting model, we innovatively introduce a self-supervised learner to enhance the forecasting model's ability to capture the spatial-temporal dynamics of traffic data and improve the generalization ability of the forecasting model. Extensive experiments on three traffic datasets under three settings prove the efficiency, effectiveness and robustness of SSTBAN.

Acknowledgment

This work was supported by the Talent Fund of Beijing Jiaotong University (Grant No. 2022RC025) and China Postdoctoral Science Foundation (Grant No. 2021M700365).

References

[1] Z. Wang, X. Su, and Z. Ding, "Long-term traffic prediction based on lstm encoder-decoder architecture," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[2] Y. Hou, P. Edara, and C. Sun, "Traffic flow forecasting for urban work zones," *IEEE transactions on intelligent transportation systems*, vol. 16, no. 4, pp. 1761–1770, 2014.

[3] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[4] L. Qu, W. Li, W. Li, D. Ma, and Y. Wang, "Daily long-term traffic flow forecasting based on a deep neural network," *Expert Systems with Applications*, vol. 121, pp. 304–312, 2019.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[6] 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*, 2021.

[7] X. Wu, D. Zhang, C. Guo, C. He, B. Yang, and C. S. Jensen, "AutoCTS: Automated correlated time series forecasting," *Proc. VLDB Endow.*, vol. 15, no. 4, pp. 971–983, 2021.

[8] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[9] 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.

[10] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv preprint arXiv:2111.06377*, 2021.
[11] N. Kim, J. Song, S. Lee, J. Choe, K. Han, S. Park, and S. Kim, "APOTS: A model for adversarial prediction of traffic speed," in *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 2022, pp. 3353–3359.
[12] H. Yuan, G. Li, Z. Bao, and L. Feng, "An effective joint prediction model for travel demands and traffic flows," in *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 2021, pp. 348–359.
[13] 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.
[14] Z. Lu, C. Zhou, J. Wu, H. Jiang, and S. Cui, "Integrating granger causality and vector auto-regression for traffic prediction of large-scale wlans." *KSII Transactions on Internet & Information Systems*, vol. 10, no. 1, 2016.
[15] H. Lee, C. Park, S. Jin, H. Chu, J. Choo, and S. Ko, "An empirical experiment on deep learning models for predicting traffic data," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 1817–1822.
[16] Z. Shao, Z. Zhang, W. Wei, F. Wang, Y. Xu, X. Cao, and C. S. Jensen, "Decoupled dynamic spatial-temporal graph neural network for traffic forecasting," *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 2733–2746, 2022.
[17] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *ICLR*, 2017.
[18] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *Advances in Neural Information Processing Systems, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
[19] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *IJCAI*, 2018.
[20] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *IJCAI*, 2019.
[21] 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, 2019, pp. 922–929.
[22] 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*, 2020.
[23] L. Han, B. Du, L. Sun, Y. Fu, Y. Lv, and H. Xiong, "Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 547–555.
[24] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.
[25] R.-G. Cirstea, B. Yang, C. Guo, T. Kieu, and S. Pan, "Towards spatio-temporal aware traffic time series forecasting," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 2900–2913.
[26] Y. Liang, K. Ouyang, J. Sun, Y. Wang, J. Zhang, Y. Zheng, D. Rosenblum, and R. Zimmermann, "Fine-grained urban flow prediction," in *Proceedings of the Web Conference 2021*, 2021, pp. 1833–1845.
[27] C. Feichtenhofer, H. Fan, Y. Li, and K. He, "Masked autoencoders as spatiotemporal learners," *arXiv preprint arXiv:2205.09113*, 2022.
[28] X. Chen, M. Ding, X. Wang, Y. Xin, S. Mo, Y. Wang, S. Han, P. Luo, G. Zeng, and J. Wang, "Context autoencoder for self-supervised representation learning," *arXiv preprint arXiv:2202.03026*, 2022.
[29] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3744–3753.
[30] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 3837–3845.
[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
[32] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," *arXiv preprint arXiv:1801.02143*, 2018.
[33] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia, "Freeway performance measurement system. mining loop detector data," *Transportation research record*, no. 1748, pp. 96–102, 2001.
[34] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," *Modeling Financial Time Series with S-Plus®*, pp. 385–429, 2006.