BigST: Linear Complexity Spatio-Temporal Graph Neural Network for Traffic Forecasting on Large-Scale Road Networks

Jindong Han
The Hong Kong University of Science and Technology
jhanao@connect.ust.hk

Weijia Zhang
The Hong Kong University of Science and Technology (Guangzhou)
wzhang411@connect.hkust-gz.edu.cn

Hao Liu
The Hong Kong University of Science and Technology (Guangzhou)
The Hong Kong University of Science and Technology
liuh@ust.hk

Tao Tao
Naiqiang Tan
Didichuxing Co. Ltd
taotao@didiglobal.com
tannaiqiang@didiglobal.com

Hui Xiong
The Hong Kong University of Science and Technology (Guangzhou)
The Hong Kong University of Science and Technology
xionghui@ust.hk

ABSTRACT

Spatio-Temporal Graph Neural Network (STGNN) has been used as a common workhorse for traffic forecasting. However, most of them require prohibitive quadratic computational complexity to capture long-range spatio-temporal dependencies, thus hindering their applications to long historical sequences on large-scale road networks in the real-world. To this end, in this paper, we propose BigST, a linear complexity spatio-temporal graph neural network, to efficiently exploit long-range spatio-temporal dependencies for large-scale traffic forecasting. Specifically, we first propose a scalable long sequence feature extractor to encode node-wise long-range inputs (*e.g.*, thousands of time-steps in the past week) into low-dimensional representations encompassing rich temporal dynamics. The resulting representations can be pre-computed and hence significantly reduce the computational overhead for prediction. Then, we build a linearized global spatial convolution network to adaptively distill time-varying graph structures, which enables fast runtime message passing along spatial dimensions in linear complexity. We empirically evaluate our model on two large-scale real-world traffic datasets. Extensive experiments demonstrate that BigST can scale to road networks with up to one hundred thousand nodes, while significantly improving prediction accuracy and efficiency compared to state-of-the-art traffic forecasting models.

PVLDB Reference Format:
Jindong Han, Weijia Zhang, Hao Liu, Tao Tao, Naiqiang Tan, and Hui Xiong. BigST: Linear Complexity Spatio-Temporal Graph Neural Network for Traffic Forecasting on Large-Scale Road Networks. PVLDB, 17(5): 1081 - 1090, 2024.
doi:10.14778/3641204.3641217

Proceedings of the VLDB Endowment, Vol. 17, No. 5 ISSN 2150-8097.
doi:10.14778/3641204.3641217

Hao Liu is the corresponding author.

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/usail-hkust/BigST.

1 INTRODUCTION

Traffic forecasting has emerged as an indispensable component of modern intelligent transportation systems with the goal of predicting future traffic dynamics (*e.g.*, flow, speed) based on historical observations [14, 22]. Accurate and timely traffic forecasting plays a significant role in congestion management and route guidance. In recent years, Spatio-Temporal Graph Neural Networks (STGNNs) [15] have shown great promise in learning spatio-temporal dependencies by incorporating the latent graph structure of the road network as an inductive bias. However, existing STGNNs still have two significant limitations when solving the traffic forecasting problem.

First, future traffic states yield complicated temporal dependencies on long-term historical observations [8, 37], *e.g.*, recurring patterns, and taking them into account is beneficial for accurate forecasting. Nevertheless, the computational and memory cost of STGNNs grows explosively as the input sequence length increases, especially when dealing with large-scale road networks. To reduce computational overhead, most existing models [1, 19, 41] only rely on the historical information within a short time window (*e.g.*, the past one hour) to make predictions, which greatly limits their performance. Some studies attempt to model long-term correlations using hand-crafted features [12, 13], but they are heavily depend on empirical assumptions, such as daily or weekly periodicity, to capture simple periodic effects. These methods are unable to model complex temporal dependencies beyond simplified assumptions.

Another essential problem inherent in STGNNs is the topology structure of the target system, which directly decides the effectiveness and efficiency of spatial dependency modeling [21]. To define the graph topology, early studies simply compute the pairwise similarity between nodes using some predefined metrics, such as road network distance [21, 44]. However, in real-world scenarios, the prior topological information of the target traffic system is usually incomplete, noisy, and may be biased towards a particular

task [1, 41]. A more effective solution is to learn a latent graph structure end-to-end under the supervision of downstream forecasting tasks. For example, GWNET [41] learns a dense adjacency matrix $\mathbf{A} = \sigma(\mathbf{E}_1\mathbf{E}_2^\top)$ to capture implicit node interactions through trainable node embedding matrices $\mathbf{E}_1$ and $\mathbf{E}_2$. Such methods reduce reliance on prior knowledge and are capable of identifying the optimal topology to facilitate learning better node representations, thus obtaining state-of-the-art performance. However, learning graph structures requires $O(N^2)$ computational complexity, where N denote the node number, which hinders their application to large-scale road networks.

This paper aims to improve the effectiveness and scalability of STGNNs in learning spatio-temporal dependencies on long historical sequences and large-scale road networks. However, two major challenges need to be addressed in achieving this goal: (1) *How to efficiently and effectively exploit knowledge over long-term historical time series?* With the increase of input sequence length, the time and space complexity grows dramatically and the model can be easily overwhelmed by noises, which poses a challenge to optimize both the model's efficiency and effectiveness. (2) *How to reduce the expensive quadratic complexity in learning latent graph structures?* As previously mentioned, existing STGNNs rely on an adaptive graph to capture complicated spatial dependencies between arbitrary nodes in road networks. However, learning a latent graph structure requires prohibitive quadratic computational complexity, which is challenging to scale to huge road networks.

To tackle the above challenges, we propose BigST, a linear complexity spatio-temporal graph neural network for large-scale traffic forecasting. BigST decouples long historical sequence modeling and traffic forecasting, treating the former as a pre-processing step. In the pre-processing step, we develop a scalable *Long Sequence Feature Extractor* (LSFE), including (1) a pre-trained context-aware linearized Transformer that encodes long-range inputs into low-dimensional representations containing rich temporal dynamics, and (2) a training-free periodic feature sampling module that explicitly captures regular periodic patterns. Note that the output of LSFE can be pre-processed to enable efficient utilization, thus addressing the first challenge. Based on the pre-processed features and the latest traffic observations, we build a *Linearized Global Spatial Convolution Network* (LGSCN) for prediction. The LGSCN first captures the time-varying graph structure of the road networks through a patch-level dynamic graph learning block, and then performs linear complexity message passing along the graph structure with an efficient spatial convolution operator, which is able to scale to large road networks.

To the best of our knowledge, BigST is one of the earliest attempts to build an efficient STGNN architecture for large-scale traffic forecasting. Our contributions are summarized as follows:

- We develop a long sequence feature extractor to provide pre-processed long-term feature representations for efficient utilization, which significantly reduces the computational overhead caused by involving long historical sequence.
- We introduce a linearized global spatial convolution network, which reduces the computational complexity to linearity for both latent graph structure learning and spatial message passing on large-scale road networks.
- Extensive experiments on two large-scale traffic datasets demonstrate the effectiveness and efficiency of the proposed model against state-of-the-art baselines. Notably, our model can scale to road networks with up to 100 thousand nodes, two orders of magnitude larger than traffic datasets commonly used in existing literature.

2 PRELIMINARIES

In this section, we introduce some basic notations and formally define the traffic forecasting problem.

Let $\mathbf{X}_t = (\mathbf{x}_t^1, \mathbf{x}_t^2, \cdots, \mathbf{x}_t^N) \in \mathbb{R}^{N\times F}$ denotes the spatio-temporal observations at time step t, where N is the number of nodes (*e.g.*, traffic sensors, road segments), $\mathbf{x}_t^i$ represents the F-dimensional time-varying traffic states (*e.g.*, traffic flow, traffic speed) of node i. We use $\mathbf{X}_{t:t+T}$ to indicate the sequence of T observations measured in the time interval $(t, t+T]$ for all the nodes. Likewise, we denote $\mathbf{U}_t$ as the external features (*e.g.*, time of day, day of week) associated to each node at time step t. The graph structure information is represented by a, possibly time-evolving, weighted adjacency matrix $\mathbf{A}_t \in \mathbb{R}^{N\times N}$. Specifically, the adjacency matrix can be derived using either a predefined metric such as distance of nodes or adaptively learned from data in an end-to-end manner. Let $\mathcal{G}_t = \langle \mathbf{X}_t, \mathbf{U}_t, \mathbf{A}_t \rangle$ indicates the spatio-temporal graph signal at time step t, we define the problem as follows.

Traffic forecasting. Given a sequence of historical T observations $\mathcal{G}_{t_0-T+1:t_0}$, the goal of traffic forecasting is to predict traffic states for all the nodes over next T_f time steps,

$$\hat{\mathbf{X}}_{t_0+1:t_0+T_f} = f_\theta(\mathcal{G}_{t_0-T+1:t_0}) \tag{1}$$

where $\hat{\mathbf{X}}_{t_0+1:t_0+T_f} = \{\hat{\mathbf{X}}_{t_0+1}, ..., \hat{\mathbf{X}}_{t_0+T_f}\}$ is the predicted traffic states from time step t_0+1 to t_0+T_f, $f_\theta(\cdot)$ denotes the machine learning-based forecasting model parameterized by θ. Notice that, we only consider a fixed size set of nodes for simplicity.

3 METHODOLOGY

3.1 Model Pipeline

As shown in Figure 1, BigST decomposes the end-to-end STGNN training into two stages: (1) pre-processing; (2) prediction and training. At pre-processing stage, we leverage a Long Sequence Feature Extractor (LSFE) trained by generative pre-training strategy [30] to efficiently pre-process the long historical traffic time series into compact vector representations, which dramatically reduce the heavy computations in prediction and training stage. At prediction and training stage, we introduce a Linearized Global Spatial Convolution Network (LGSCN), which consists of a Patch-level Dynamic Graph Learning (PDGL) block and multiple stacked Linearized Spatial Convolution (LSC) layers. The PDGL block first quantifies the dynamic spatial dependencies by utilizing an attention-based scoring neural network, and then constructs the graph structures based on the derived attention scores. The LSC enables fast spatial message passing along the learned graph structure by avoiding the explicit computation of dense adjacency matrix, leading to linear time and space complexity *w.r.t.* the number of input nodes.

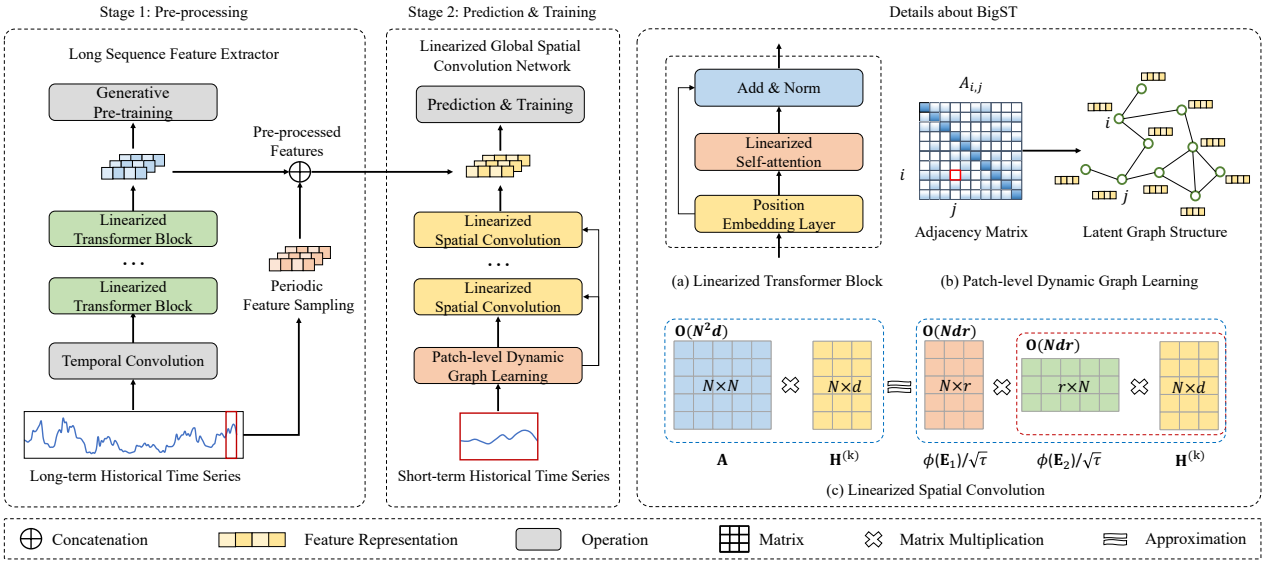


Figure 1: The model pipeline of BigST.

3.2 Long Sequence Feature Extractor

To effectively and efficiently extract knowledge from long temporal sequence inputs, we first introduce the *Long Sequence Feature Extractor* (LSFE), which includes two modules: (1) a context-aware linearized Transformer to model complicated interactions between distant positions in time series with linear complexity, and (2) a periodic feature sampling module to explicitly capture periodic traffic patterns through hand-crafted features.

3.2.1 ***Context-aware linearized Transformer.*** The major challenge of modeling long sequence is to reduce signal traveling path between distant positions in time series while maintaining low complexity [48]. Recently Transformer [36] significantly shortens the maximum length of signal traveling path to $O(1)$, but suffers from quadratic time and space complexity, thus restricting their applicability in long temporal sequences. We address this issue by employing a context-aware linearized Transformer, which is described as follows.

Formally, given a sequence of historical traffic observations $\mathbf{x}^i_{t_0-T_l:t_0} = \left\{\mathbf{x}^i_{t_0-T_l}, \mathbf{x}^i_{t_0-T_l+1}, \ldots, \mathbf{x}^i_{t_0}\right\}$ for node i, where T_l denotes the size of time window, we aim to capture the dependencies among different time steps. However, the semantics of a single time point is much lower than that of a word in a sentence, which could degrade the learning capability [29, 33]. To overcome this limitation, we capture local context information that is not available in point-wise values through a temporal convolution layer [45]. The convolution operator at time step t is defined as follows

$$\mathbf{x}^i \star f(t) = \sum_{s=0}^{S-1} f(s) \cdot \mathbf{x}^i_{t-\omega\times s}, \tag{2}$$

where ω is the dilation factor, $f(\cdot)$ is the filter kernel with size S. In practice, we can leverage a set of filters to extract rich context features $\mathbf{c}^i_t$ for each time step t.

Afterward, we adopt multiple linearized Transformer blocks to encode rich long-term temporal dynamics. Each block includes a position embedding layer and a linearized self-attention layer. Specifically, we first inject position encoding into $\mathbf{c}^i_t$ to preserve ordering information in the time series, denoted as $\mathbf{h}^i_t = \mathbf{c}^i_t + \mathbf{p}_t$, where $\mathbf{p}_t$ represents a learnable temporal embedding for time step t. Here we omit superscript i for simplicity. We then project the output $\mathbf{h}_t$ into three types of vectors separately, defined as

$$\mathbf{q}_t = \mathbf{W}_q\mathbf{h}_t, \mathbf{k}_t = \mathbf{W}_k\mathbf{h}_t, \mathbf{v}_t = \mathbf{W}_k\mathbf{h}_t, \tag{3}$$

where $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ are learnable parameters. Following common terminology [36], we employ *query*, *key*, and *value* to denote $\mathbf{q}_t$, $\mathbf{k}_t$, and $\mathbf{v}_t$, respectively. After obtaining queries, keys, and values, we use the self-attention mechanism to capture latent temporal dependencies and enable information exchange between arbitrary time steps t and m

$$\mathbf{h}'_t = \sum_{m=t_0-T_l}^{t_0} \frac{\exp((\mathbf{q}_t)^\top \mathbf{k}_m)}{\sum_{n=t_0-T_l}^{t_0} \exp((\mathbf{q}_t)^\top \mathbf{k}_n)} \cdot \mathbf{v}_m, \tag{4}$$

where $\mathbf{h}'_t$ is the output of self-attention at time step t. Nevertheless, such operation suffers from quadratic time and space complexity (*i.e.*, $O(T_l^2)$) when dealing with long historical time series. To efficiently capture long-range temporal dependencies while achieving linear complexity, we further utilize the linearized self-attention, which is described below.

Concretely, we define a Softmax kernel function $k(\mathbf{q}_t, \mathbf{k}_m) = \exp((\mathbf{q}_t)^\top \mathbf{k}_m)$. In general, any positive definite kernel function can be approximated using a randomized feature map $\phi: \mathbb{R}^d \to \mathbb{R}^r_+$ with unbiased estimation [31], defined as $k(\mathbf{q}_t, \mathbf{k}_m) \approx \phi(\mathbf{q}_t)^\top \phi(\mathbf{k}_m)$. Similarly, we can borrow the idea of Performer [3] and leverage Positive Random Features (PRFs) for Softmax kernel approximation

$\exp((\mathbf{q}_t)^\top \mathbf{k}_m) \approx \phi(\mathbf{q}_t)^\top \phi(\mathbf{k}_m)$, where $\phi(\mathbf{x})$ is defined as

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{r}}[\exp(\mathbf{w}_1^\top \mathbf{x} - \frac{\|\mathbf{x}\|_2^2}{2}), \cdots, \exp(\mathbf{w}_r^\top \mathbf{x} - \frac{\|\mathbf{x}\|_2^2}{2})], \quad (5)$$

where $\mathbf{w}_r \sim \mathcal{N}(0, I_d)$. Given $\phi(\mathbf{q}_t)$ and $\phi(\mathbf{k}_m)$, equation 4 can be estimated as follows

$$\mathbf{h}'_t = \frac{\phi(\mathbf{q}_t)^\top \sum_{m=t_0-T_l}^{t_0} \phi(\mathbf{k}_m) \cdot \mathbf{v}_m}{\phi(\mathbf{q}_t)^\top \sum_{n=t_0-T_l}^{t_0} \phi(\mathbf{k}_n)}, \quad (6)$$

Note that $\mathbf{a}_{t_0} = \sum_{m=t_0-T_l}^{t_0} \phi(\mathbf{k}_m) \cdot \mathbf{v}_m$ and $\mathbf{b}_{t_0} = \sum_{n=t_0-T_l}^{t_0} \phi(\mathbf{k}_n)$ are shared across different time steps. Therefore, we can reuse them for each query, resulting in $O(T_l)$ complexity.

On top of the context-aware linearized Transformer, we average the hidden representations of the last K time steps into $\mathbf{h}_{long}$ and employ a fully connected layer to project $\mathbf{h}_{long}$ to the traffic states of next T_f time steps $\{\hat{\mathbf{x}}_{t_0+1}, ..., \hat{\mathbf{x}}_{t_0+T_f}\}$. Finally, we pre-train the Transformer model through generative pre-training strategy [30] by optimizing the following objective

$$\mathcal{L}_s = \frac{1}{T_f} \sum_{t=t_0+1}^{t=t_0+T_f} |\mathrm{x}_t - \hat{\mathbf{x}}_t|, \quad (7)$$

where x_t is the ground-truth traffic state at time t. As spatial correlations are not considered in LSFE, the model can be efficiently pre-trained by uniformly sampling mini-batches of node samples.

3.2.2 ***Periodic feature sampling.*** In addition, urban traffic usually exhibits strong daily and weekly periodicity [37, 46]. To effectively represent various types of periodic patterns, we construct a rich pool of hand-crafted features. Specifically, suppose we aim to make prediction for future time interval $(t_0, t_0 + T_f]$, we sample traffic state features from the same period in the past D days and W weeks and aggregate them into a feature vector h_{pe}.

3.2.3 ***Pre-processing.*** We can utilize the pre-trained context-aware linearized Transformer to generate pre-computed representations by pre-processing the entire dataset, which significantly reduces the computational burden at the prediction stage. Specifically, we denote the output of context-aware linearized Transformer and periodic feature sampling as H_{long} and H_{pe} with rows given by h_{long} and h_{pe}, respectively. H_{long} and H_{pe} can be further incorporated into the subsequent linearized global spatial convolution network as additional inputs to improve performance, which we will discuss in Section 3.3.3.

3.3 Linearized Global Spatial Convolution Network

Existing studies [1, 40, 41] mainly exploit an adaptive graph learned from data to capture dependencies among different traffic sensors. The inferred graph structure is often static and cannot capture the time-varying node dependence that commonly exists in practical scenarios [27, 43]. Moreover, the computational overhead for learning a latent graph structure grows quadratically with the number of nodes N, which is prohibitively expensive for handling large road networks. The problem is even more dramatic when graph convolution is applied for feature aggregation at each time step, leading to the complexity $O(TLN^2)$, where T and L denote the number of time steps and graph convolution layers, respectively. Previous studies [40] rely on node subset sampling to reduce the heavy computations in training phase. However, sampling a subset of nodes may disrupt the underlying spatio-temporal dependencies, leading to information loss and performance degradation.

To address the above issues, we propose a *Linearized Global Spatial Convolution Network* (LGSCN), which includes (1) a patch-level dynamic graph learning block to capture the time-varying correlations among arbitrary node pairs, and (2) a linearized spatial convolution block that can scale to large road networks without perturbing any spatio-temporal dependencies.

3.3.1 ***Patch-level dynamic graph learning.*** To reduce the computational overhead, the LGSCN module only takes the latest T observations as input, where T is a small value, *e.g.*, 12 time steps. Specifically, given previous T steps' node observations $\mathbf{x}^i_{t_0-T:t_0} = \{\mathbf{x}^i_{t_0-T}, \mathbf{x}^i_{t_0-T+1}, \ldots, \mathbf{x}^i_{t_0}\}$, we first aggregate $\mathbf{x}^i_{t_0-T:t_0}$ into a subseries-level patch using the concatenate operation $||$, defined as

$$\mathbf{x}^i = \mathbf{x}^i_{t_0-T} || \mathbf{x}^i_{t_0-T+1} || ... || \mathbf{x}^i_{t_0}. \quad (8)$$

The patch-wise input has a more informative representation than point-wise input, making model learning easier [29]. Moreover, by considering the time series as a whole, we can avoid performing graph convolution at each time step, which significantly reduces the computational and storage cost.

To quantify the time-varying node dependence, we maintain two embedding vectors for each node i, *i.e.*, a static embedding $\overline{\mathbf{u}}^i$ and a dynamic embedding $\mathbf{u}^i$. The static embedding $\overline{\mathbf{u}}^i$ can be encoded by learnable parameters and does not change over time, representing inherent node properties. The dynamic embedding $\mathbf{u}^i = \mathbf{x}^i || \mathbf{s}^h || \mathbf{s}^w$ captures short-term evolving patterns, where $\mathbf{s}^h$ and $\mathbf{s}^w$ are the learnable embeddings of discrete temporal features, *i.e.*, time of day and day of week. Based on $\overline{\mathbf{u}}^i$ and $\mathbf{u}^i$, we compute the attention score between node i and j via dot product

$$\alpha_{i,j} = (\mathbf{e}_1^i)^\top \mathbf{e}_2^j, \quad (9)$$

$$\mathbf{e}_1^i = \mathbf{W}_1[\overline{\mathbf{u}}^i || \mathbf{u}^i], \mathbf{e}_2^i = \mathbf{W}_2[\overline{\mathbf{u}}^i || \mathbf{u}^i], \quad (10)$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ represent the learnable projection matrices. Attention score $\alpha_{i,j}$ is utilized for graph construction, we define the edge weight as follows

$$A_{i,j} = \frac{\exp(\alpha_{i,j}/\tau)}{\sum_{k=1}^{N} \exp(\alpha_{i,k}/\tau)}, \quad (11)$$

where $A_{i,j}$ denotes the edge weight between node i and j, τ is the temperature coefficient. A smaller τ can potentially amplify the impact of strongly connected nodes while eliminating weak connections, which further improves the performance.

3.3.2 ***Linearized spatial convolution.*** To enable information exchange among nodes, we introduce a mix-hop feature propagation operator as follows

$$\mathbf{H}^{(0)} = \overline{\mathbf{U}} || \mathbf{U}, \quad (12)$$

$$\mathbf{H}^{(k)} = \mathbf{A}\mathbf{H}^{(k-1)}\mathbf{W}^{(k-1)}, \quad (13)$$

where the i-th row vectors of $\overline{\mathbf{U}}$ and $\mathbf{U}$ are $\overline{\mathbf{u}}^i$ and $\mathbf{u}^i$ respectively, K is the depth of feature propagation, $\mathbf{W}^{(k)}$ is a learnable weighted matrix, $\mathbf{H}^{(k)}$ denotes the node representations at hop k, $\mathbf{A}$ is the

adjacency matrix derived by Equation 11. After K-hop feature propagation, we can obtain a series of node representations at different hops: $\{\mathbf{H}^{(0)}, \mathbf{H}^{(1)}, \ldots, \mathbf{H}^{(K)}\}$. We further aggregate all the node representations as follows

$$\mathbf{H} = AGG(\mathbf{H}^{(0)}, \mathbf{H}^{(1)}, \ldots, \mathbf{H}^{(K)}), \tag{14}$$

where $AGG(\cdot)$ represents an aggregator function, which is instantiated by a concatenate operation in practice. Unfortunately, the calculation of adjacency matrix $\mathbf{A}$ and the feature propagation for N nodes requires $O(N^2)$ complexity, which scales poorly as the number of nodes increases. Inspired by recent advancements in graph Transformer [38], we develop a linearized spatial convolution operator to achieve efficient feature propagation. Concretely, we can rewrite the computation of adjacency matrix $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{D}^{-1}\mathbf{A}', \tag{15}$$

$$\mathbf{A}' = \exp(\mathbf{E}_1\mathbf{E}_2^\top/\tau), \mathbf{D} = \mathrm{diag}(\mathbf{A}'\mathbf{1}_N), \tag{16}$$

where the i-th row vectors of $\mathbf{E}_1$ and $\mathbf{E}_2$ are $\mathbf{e}_1^i$ and $\mathbf{e}_2^i$ respectively, $\mathbf{1}_N$ is the all-ones vector with N dimension, and $\mathrm{diag}(\cdot)$ transforms the input vector into a diagonal matrix.

Here we reuse Equation 5 to factorize the adjacency matrix $\mathbf{A}$. For $\phi(\mathbf{E}_1)$ and $\phi(\mathbf{E}_2)$ with rows given by $\phi(\mathbf{e}_1^i)$ and $\phi(\mathbf{e}_2^i)$, Equation 15 can be approximated as follows

$$\mathbf{A} \approx \mathbf{D}^{-1}(\phi(\mathbf{E}_1)\phi(\mathbf{E}_2)^\top/\tau) = \mathbf{D}^{-1}\hat{\mathbf{E}_1}\hat{\mathbf{E}_2}^\top, \tag{17}$$

where $\hat{\mathbf{E}_1} = \phi(\mathbf{E}_1)/\sqrt{\tau}$ and $\hat{\mathbf{E}_2} = \phi(\mathbf{E}_2)/\sqrt{\tau}$. Finally, we accelerate Equation 13 with the following linearized feature propagation

$$\mathbf{H}^{(k)} = \hat{\mathbf{D}}^{-1}(\hat{\mathbf{E}_1}(\hat{\mathbf{E}_2}^\top\mathbf{H}^{(k-1)}))\mathbf{W}^{(k-1)}, \tag{18}$$

$$\hat{\mathbf{D}} = \mathrm{diag}(\hat{\mathbf{E}_1}(\hat{\mathbf{E}_2}^\top\mathbf{1}_N)), \tag{19}$$

where the brackets represent the order of matrix multiplication. With the above operator, we do not need to explicitly materialize the quadratic adjacency matrix $\mathbf{A}$. It is obvious that the linearized spatial convolution have smaller time and space complexity (*i.e.*, $O(N)$) than regular one.

Moreover, we also impose prior spatial constraints on the learned graph structure by introducing a additional loss function

$$\mathcal{L}_r = \sum_{ij} -d_{i,j}\log A_{i,j}, \tag{20}$$

where $d_{i,j}$ indicates the spatial proximity among node i and j in the road network, *e.g.*, road connectivity.

3.3.3 ***Prediction and training.*** At prediction and training stage, we concatenate the output of LGSCN $\mathbf{H}$, the pre-processed feature representations $\mathbf{H}_{long}$ and $\mathbf{H}_{per}$ derived from LSFE, and then adopt a Multi-Layer Perceptron (MLP) Φ_{mlp} to generate the final prediction results $\hat{\mathbf{X}}_{t_0+1:t_0+T_f} = \Phi_{mlp}(\mathbf{H}||\mathbf{H}_{long}||\mathbf{H}_{per})$. Notice that, to accelerate the inference speed, we directly project the hidden representation to the final output $\hat{\mathbf{X}}_{t_0+1:t_0+T_f} \in \mathbb{R}^{N\times T_f}$ in a non-autoregressive way. The LGSCN is trained by optimizing the following objective:

$$\mathcal{L}_p = \frac{1}{NT_f}\sum_{i=t_0+1}^{i=t_0+T_f}|\mathbf{X}^i - \hat{\mathbf{X}}^i| + \lambda\mathcal{L}_r \tag{21}$$

where λ controls the importance of prior spatial knowledge.

Table 1: Statistics of datasets.

Datasets	California	Beijing
#Nodes	9,638	99,716
#Time Steps	28,224	10,944
#Traffic Records	272,022,912	1,091,291,904
Time Span	3 months	1 month
Variable	Speed	Speed

3.4 Complexity Analysis

In this section, we analyze the algorithmic complexity of each proposed module. We first denote the number of input nodes, the number of time steps in LSFE, the propagation depth, the hidden dimensionality, and the dimension of random features as N, T_l, K, d, and r, respectively.

For the LSFE module, the time and space complexity of dilated causal convolution are $O(T_l d^2/\omega)$ and $O(T_l d)$, where ω denotes dilation factor. The time and space complexity of a single linearized Transformer block are $O(T_l dr)$ and $O(T_l r)$. If we stack L Transformer blocks, the overall time complexity becomes $O(LT_l dr)$ and space complexity becomes $O(LT_l r)$. Additionally, since the periodic feature sampling module is training-free, we omit its time and memory cost. For the LGSCN module, the original time and space complexity are $O(KNd^2 + N^2 d)$ and $O(KNd + N^2)$, respectively. Fortunately, we can reduce the complexity to $O(KNd^2 + Ndr)$ and $O(KNd + Nr)$ via spatial convolution approximation, as we only need to compute and store matrices $\hat{\mathbf{E}_1}$ and $\hat{\mathbf{E}_2}$. Overall, the model complexity has linear dependency on the sequence length T_l and node number N, making it efficient and scalable to deal with long historical time series and large-scale road networks.

4 EXPERIMENT

4.1 Experimental Setup

4.1.1 ***Data description.*** We conduct experiments on two real-world traffic datasets, California and Beijing. The California dataset contains traffic data derived from the Caltrans Performance Measurement System (PeMS)[1], which has been widely used for evaluating traffic forecasting models [4, 39, 41]. However, unlike previous studies, we utilize the full sensor network consisting of 9,638 traffic detectors distributed across nine districts for model evaluation. Additionally, the Beijing dataset is collected from Didi, a large-scale ride-hailing platform. This dataset contains traffic speed measurements calculated from GPS devices over 99,716 road segments in Beijing city. All traffic datasets are aggregated every 5 minutes, resulting in 288 time slots per day. We split the datasets into training, validation, and test data by using the ratio of 7:1:2. We apply Z-score normalization to normalize the input traffic data for model training. We also construct an additional prior adjacency matrix for each dataset by considering the actual distance and connectivity among sensors and road segments. We summarize the statistics of each dataset in Table 1.

4.1.2 ***Baselines.*** We compare the proposed approach with the following typical baselines: (1) **Historical Average (HA)** predicts

[1] https://pems.dot.ca.gov/

Table 2: Overall performance comparison evaluated by MAE, MAPE, RMSE on California and Beijing with different horizons. We use graph partitioning algorithm [16] to avoid Out-Of-Memory (OOM) issue for memory-heavy baseline models.

Dataset	Algorithm	Horizon 3			Horizon 6			Horizon 12			Average		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
California	HA	1.79	3.56%	3.63	2.10	4.23%	4.22	2.61	5.36%	5.13	2.12	4.28%	4.24
	VAR	1.56	3.05%	2.95	1.91	3.86%	3.66	2.35	5.01%	4.51	1.97	3.98%	3.64
	DCRNN	1.34	2.53%	2.70	1.71	3.40%	3.55	2.21	4.63%	4.48	1.68	3.38%	3.45
	ASTGCN	1.44	2.77%	2.90	1.77	3.57%	3.65	2.23	4.79%	4.49	1.76	3.57%	3.55
	GWNET	1.26	2.39%	2.60	1.57	3.17%	3.34	1.94	4.16%	4.09	1.52	3.12%	3.23
	AGCRN	1.31	2.55%	2.62	1.61	3.28%	3.31	1.98	4.22%	4.02	1.58	3.23%	3.21
	STGODE	1.35	2.65%	2.67	1.66	3.38%	3.37	2.05	4.41%	4.16	1.63	3.35%	3.28
	DSTAGNN	1.38	2.72%	2.73	1.69	3.44%	3.42	2.12	4.48%	4.24	1.68	3.42%	3.35
	BigST	**1.24**	**2.37%**	**2.49**	**1.48**	**3.01%**	**3.09**	**1.71**	**3.61%**	**3.56**	**1.43**	**2.90%**	**2.96**
Beijing	HA	19.47	31.63%	27.79	20.38	32.92%	28.44	21.17	34.57%	30.03	19.87	32.16%	28.09
	VAR	14.78	21.95%	22.08	15.57	23.61%	22.86	16.32	24.95%	23.89	15.25	22.99%	22.61
	DCRNN	12.07	17.43%	20.48	12.66	18.41%	21.44	13.62	19.84%	22.24	12.52	18.33%	21.05
	ASTGCN	14.28	20.97%	21.40	15.08	22.70%	22.36	15.85	23.85%	23.11	14.74	21.91%	21.90
	GWNET	11.01	16.71%	19.95	12.03	18.14%	21.29	12.55	19.25%	21.88	11.49	17.53%	20.55
	AGCRN	11.26	16.69%	19.85	12.22	18.11%	21.23	12.79	19.37%	21.89	11.76	17.62%	20.53
	STGODE	11.89	17.40%	20.42	12.70	18.64%	21.79	13.25	19.78%	22.02	12.28	18.09%	20.87
	DSTAGNN	13.81	20.14%	20.92	14.96	22.41%	22.36	15.54	23.23%	22.69	14.37	21.23%	21.49
	BigST	**10.39**	**16.13%**	**19.22**	**11.04**	**17.72%**	**20.52**	**11.32**	**18.51%**	**20.81**	**10.51**	**16.82%**	**19.76**

future traffic states by leveraging the average of historical observations from the same time slot in previous seasons (*e.g.*, previous days). (2) **Vector Auto-Regression (VAR)** [28] is a statistical model that can capture the inter-dependency relationships of multivariate time series. In this work, we directly implement VAR for traffic forecasting. (3) **DCRNN** [21] models traffic as a diffusion process and captures spatio-temporal dependencies via diffusion convolution operator coupled with recurrent neural networks. (4) **ASTGCN** [12] combines spatial and temporal attention mechanisms to capture dynamic spatio-temporal correlations for traffic forecasting. (5) **GWNET** [41] stacks multiple adaptive graph convolution and dilated casual convolution layers to capture spatio-temporal traffic dynamics at different levels. (6) **AGCRN** [1] further enhances the performance of traffic forecasting by capturing node-specific patterns through node adaptive graph convolution. (7) **STGODE** [9] captures node relationships via both road network distance and semantic similarity, and adopts a tensor-based ordinary differential equation for traffic modeling. (8) **DSTAGNN** [19] improves traffic forecasting performance by utilizing spatio-temporal aware graph inferred from data and multi-scale gated convolution.

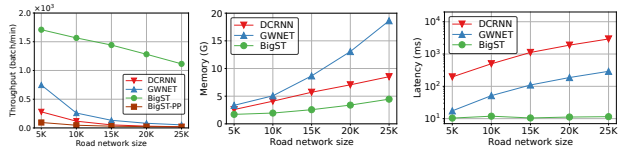
4.1.3 ***Evaluation metrics.*** Following previous studies [21], we use three widely adopted metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE), for model evaluation.

4.1.4 ***Implementation details.*** Our approach and all deep learning baselines are implemented with Pytorch. Specifically, we aim to predict 12-step-ahead observations, *i.e.*, the next one hour. We set the look-back time window T_l to 2016 and 864 for California and Beijing datasets, respectively. For the LSFE module, we employ 32 filters with a filter size of 6. We set the random feature dimension r to 64, and the hidden dimension of linearized Transformer is fixed to 32. Furthermore, we set D and W to 7 and 2, respectively. For the LGSCN module, we set the dimension of static embedding to 16, while fixing the dimension of dynamic embedding to 32. The hidden dimension of LGSCN is fixed to 128, and we utilize the LeakyReLU (alpha=0.2) function in our model. The temperature coefficient τ is set to 0.2. Finally, we set the dimension r of random feature map to 64 and λ to 0.3. The model is trained with the Adam optimizer, using a learning rate of 0.0001 and a gradient clip of 5.

4.2 Performance Comparison

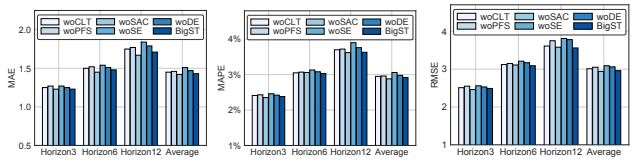
Table 2 presents the overall performance of our model and all the baselines with respect to MAE, RMSE, and MAPE on two datasets. To avoid Out-Of-Memory (OOM) issue for memory-intensive baseline models, we first divide the large road network into different sub-networks via graph partitioning algorithm [16] and then train baseline models for each sub-network.

Overall, BigST consistently outperforms other baseline models on both the California and Beijing datasets, which demonstrates the advance of the proposed model. Specifically, our approach achieves 6.3%, 7.6%, 8.4% improvement beyond the best baselines (*i.e.*, GWNET and AGCRN) in average MAE, RMSE, and MAPE on California dataset. Likewise, the improvements on Beijing dataset are 9.3%, 4.2%, 3.9%, respectively. Furthermore, we can make the following observations: (1) HA achieves the worst performance, as it overlooks the complex variable correlations in traffic data. (2) All STGNN-based approaches outperform HA and VAR by a large margin, validating the superiority of modeling nonlinear and dynamic spatio-temporal dependencies. (3) STGNNs using an adaptive adjacency matrix, such as GWNET and AGCRN,



(a) Training throughput. (b) Memory usage. (c) Inference latency.

Figure 2: Efficiency analysis. BigST-PP indicates throughput at pre-processing stage.



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

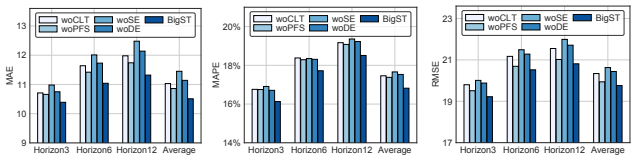
Figure 3: Ablation study on California.

exhibit strong learning capabilities and achieve better performance than STGNNs that only use pre-defined graph structures, such as DCRNN. The possible reason is that the pre-defined adjacency matrix is usually noisy and struggles with capturing the long-range spatial dependencies.

4.3 Efficiency Analysis

In this section, we conduct the experiments to examine the efficiency of the proposed module *w.r.t.* the number of nodes. Specifically, we adopt training throughput, GPU memory usage, and inference latency as evaluation metrics. In this work, throughput represents the average number of batches the model can process in one minute, *i.e.*, batch/min. The batch size is set to 1 to avoid memory overflow of baseline models. Since BigST is a two-stage training approach, we also report the throughput at pre-processing stage, marked as BigST-PP.

We compare our model with two typical baseline models, *i.e.*, DCRNN and GWNET. Figure 2 shows the training throughput, GPU memory usage, and inference latency depending on the input number of nodes. Compared to GWNET, our model achieves 2.3×~20.6× and 1.7×~26.5× higher training and inference acceleration, while reducing memory cost by up to 76.1%. Compared to DCRNN, the speedup for throughput and latency becomes more dramatic, as DCRNN employs a recurrent structure to capture temporal dependency and makes prediction in a step-by-step manner, which could significantly slow down the model training and inference. Additionally, we also notice that the inference latency of BigST remains relatively stable or even decreases slightly as the road network size increases. The possible reason is that GPUs can fully utilize its resources to process the data in parallel due to the linear complexity of our algorithm.



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

Figure 4: Ablation study on Beijing. The missing results of woSAC is due to memory overflow.

4.4 Ablation Studies

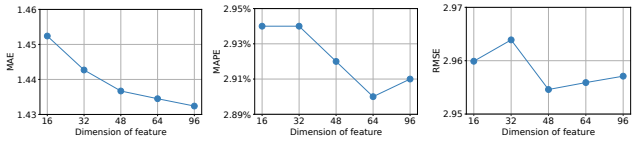
In this section, we conduct ablation studies on California and Beijing to evaluate the effectiveness of each proposed module. We test five variants: (1) *woCLT* removes the pre-computed long-term representations generated by the context-aware linearized Transformer, (2) *woPFS* excludes periodic features, (3) *woSAC* removes spatial convolution approximation, (4) *woSE* includes only dynamic embedding for graph construction, (5) *woDE* includes only static embedding. The results are shown in Figure 4.

First, a performance degradation occurs when we remove either long-term representations or periodic features, indicating the importance of modeling long-range temporal dependencies. We also observe that long-term representations have a greater impact on the Beijing dataset than periodic features. This could be due to the fact that traffic conditions in Beijing are more complex and exhibit weaker regular periodicity than in California. Second, although the use of spatial convolution approximation reduces the model complexity on large-scale road networks, it leads to a noticeable performance degradation. Third, removing dynamic or static embedding in graph construction procedure deteriorates the model performance, validating the effectiveness of both static and dynamic embeddings for learning time-varying graph structures. Moreover, static embedding has a more significant impact on model performance. This is because short-term information contains a lot of noise and is unable to capture the reliable spatial graph structure.

4.5 Parameter Sensitivity

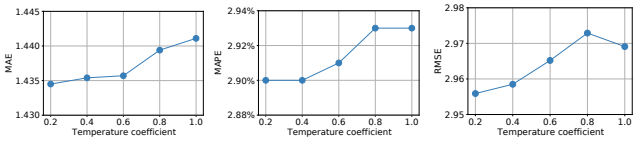
Finally, we study the impact of hyper-parameters on the performance of traffic forecasting. We evaluate the impact of the dimension of random features r and temperature coefficient τ, two critical hyper-parameters related to our contributions. We perform the parameter sensitivity analysis on California dataset, and the results on Beijing are similar. We set other hyper-parameters to their optimal values when we study the target one.

First, we vary the dimension of random features r from 16 to 96. The results on MAE, MAPE and RMSE are reported in Figure 5. We can observe a clear trend of performance improvement when increasing r from 16 to 96. More specifically, the forecasting accuracy first increases rapidly and then slightly decreases, indicating that a larger random feature dimension can provide a better approximation for the time-varying adjacency matrix. However, as increasing r introduces extra overhead, we choose $r = 64$ to reduce the computation cost while achieving reliable approximation.



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

Figure 5: Effect of random feature dimension r.



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

Figure 6: Effect of temperature coefficient τ.

Then, we vary the temperature coefficient τ from 0.2 to 1.0. The results are reported in 6. Overall, we achieve best performance when $\tau = 0.2$. The results show a performance gain when decreasing τ from 1.0 to 0.4, and the performance becomes stable by further decreasing τ from 0.4 to 0.2. This is because a smaller τ can help prune out weak connections, which avoids over-fitting spurious correlations among independent nodes.

5 RELATED WORK

5.1 Traffic Forecasting

Traffic forecasting aims to provide accurate and timely predictions of traffic dynamics, *e.g.*, traffic flow [9], traffic speed [41], and transportation demand [11, 24], which plays an essential role in modern intelligent transportation systems [23]. With the development of deep learning, many models have been proposed for traffic forecasting. For example, STDN [42] split the whole city into an image-like grid map and employ convolution neural networks (CNNs) to predict future traffic flow of each grid. However, such CNN-based method can only handle regular gridded data and fails to model the irregular graph structure of the traffic systems, e.g., traffic flows on road networks. Recently, Spatio-Temporal Graph Neural Networks (GNNs) [15] have emerged as an effective tool for graph-based traffic forecasting. Previous studies on STGNNs roughly fall into two categories. (1) *Prior-based approaches* [9, 10, 21, 44] leverage pre-defined graphs to model non-Euclidean spatial dependencies for traffic forecasting. This line of research highly relies on heuristic metrics, which are not directly related to downstream forecasting tasks. (2) *Graph learning approaches* [1, 2, 19, 33, 34, 41] address the above issue by automatically learning a latent fully-connected graph in an end-to-end fashion. For instance, GWNET [41] learns an adaptive graph topology by factorizing the adjacency matrix into two node embedding matrices. STEP [33] learns a discrete graph structure by using the long-term information extracted from a Transformer model. However, these methods suffer from quadratic complexity issue, which hinders their applications to real-world large-scale road networks. Another line of research focus on addressing the indistingushibility in spatiotemporal data to enhance forecasting accuracy [6, 32]. For example, STID incorporates spatiotemporal identity information to achieve comparable performance with STGNNs. However, they do not explicitly model spatial diffusion process, which may lead to performance degradation in complex traffic scenarios with rich neighorbood information [26].

5.2 Transformer-based Models

In recent years, Transformer-based models have achieved significant success in various applications, including computer vision [7], natural language processing [36], and graph mining [38]. The fundamental building block of the Transformer, *i.e.*, self-attention, dramatically reduces the maximum signal traversing path into $O(1)$, thus showing great potential in capturing dependencies between distant positions in long sequences. However, the canonical self-attention suffers from quadratic complexity, limiting its direct use in handling extremely long sequences. To address this issue, several techniques have been proposed, including pooling mechanisms [5], hashing methods [18], and kernel-based methods [3, 17]. Due to the powerful sequential modeling ability, Transformer-based models have gained increasing attention in time series forecasting tasks [20, 25, 29, 35, 47, 48]. To name a few, Informer [48] proposes ProbSparse self-attention with distillation techniques to efficiently capture long-range dependencies in time series. PatchTST [25] splits the time series into a series of patches before feeding them into the Transformer model. In this paper, we employ the Transformer model by using linear attention to efficiently encode long-term temporal dynamics for large-scale traffic forecasting.

6 CONCLUSION

In this paper, we introduce BigST, a linear-complexity STGNN architecture for large-scale traffic forecasting. We first develop a scalable long sequence feature extractor to encode the long historical sequences into low-dimensional representations. To achieve this, we leverage a context-aware kernelized Transformer trained with generative pre-training strategy to efficiently capture long-range temporal correlations with linear complexity. Subsequently, we construct a linearized global spatial convolution network to enable fast message passing by approximating the dense adjacency matrix using two lightweight matrices, resulting in time and space complexity that scale linearly with the number of input nodes. Extensive experimental results show that our approach achieves performance and efficiency superiority on large-scale datasets with up to 100 thousands nodes, which is two orders of magnitude larger than medium-sized datasets commonly used in current literature. In the future, we plan to deploy BigST to large-scale real-world services such as online maps and ride-hailing platforms.

ACKNOWLEDGMENTS

This research was supported in part by the National Natural Science Foundation of China under Grant No. (62102110, 92370204), Guangzhou Science and Technology Plan Guangzhou-HKUST(GZ) Joint Project No. 2023A03J0144, and the Education Bureau of Guangzhou Municipality.

REFERENCES

[1] 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.

[2] Jeongwhan Choi, Hwangyong Choi, Jeehyun Hwang, and Noseong Park. 2022. Graph neural controlled differential equations for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 6367–6374.

[3] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794* (2020).

[4] Yue Cui, Kai Zheng, Dingshan Cui, Jiandong Xie, Liwei Deng, Feiteng Huang, and Xiaofang Zhou. 2021. METRO: a generic graph neural network framework for multivariate time series forecasting. *Proceedings of the VLDB Endowment* 15, 2 (2021), 224–236.

[5] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems* 33 (2020), 4271–4282.

[6] Jinliang Deng, Xiusi Chen, Renhe Jiang, Xuan Song, and Ivor W Tsang. 2021. St-norm: Spatial and temporal normalization for multi-variate time series forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 269–278.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[8] Wei Fan, Shun Zheng, Xiaohan Yi, Wei Cao, Yanjie Fu, Jiang Bian, and Tieyan Liu. 2022. DEPTS: Deep expansion learning for periodic time series forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=AJAR-JgNw__

[9] 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 and Data Mining*. 364–373.

[10] Ziquan Fang, Lu Pan, Lu Chen, Yuntao Du, and Yunjun Gao. 2021. MDTP: A multi-source deep traffic prediction framework over spatio-temporal trajectory data. *Proceedings of the VLDB Endowment* 14, 8 (2021), 1289–1297.

[11] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3656–3663.

[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 *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.

[13] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. 2021. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering* 34, 11 (2021), 5415–5428.

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

[15] Guangyin Jin, Yuxuan Liang, Yuchen Fang, Jincai Huang, Junbo Zhang, and Yu Zheng. 2023. Spatio-Temporal Graph Neural Networks for Predictive Learning in Urban Computing: A Survey. *arXiv preprint arXiv:2303.14483* (2023).

[16] George Karypis and Vipin Kumar. 1998. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN* 38 (1998), 7–1.

[17] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*. PMLR, 5156–5165.

[18] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[19] 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.

[20] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems* 32 (2019).

[21] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[22] Fan Liu, Hao Liu, and Wenzhao Jiang. 2022. Practical adversarial attacks on spatiotemporal traffic forecasting models. *Advances in Neural Information Processing Systems* 35 (2022), 19035–19047.

[23] Hao Liu, Ying Li, Yanjie Fu, Huaibo Mei, Jingbo Zhou, Xu Ma, and Hui Xiong. 2020. Polestar: An intelligent, efficient and national-wide public transportation routing engine. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2321–2329.

[24] Hao Liu, Qiyu Wu, Fuzhen Zhuang, Xinjiang Lu, Dejing Dou, and Hui Xiong. 2021. Community-Aware Multi-Task Transportation Demand Prediction. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 320–327.

[25] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

[26] Xu Liu, Yuxuan Liang, Chao Huang, Hengchang Hu, Yushi Cao, Bryan Hooi, and Roger Zimmermann. 2023. Do We Really Need Graph Neural Networks for Traffic Forecasting? *arXiv preprint arXiv:2301.12603* (2023).

[27] Yijing Liu, Qinxian Liu, Jianwei Zhang, Haozhe Feng, Zhongwei Wang, Zihan Zhou, and Wei Chen. 2022. Multivariate Time-Series Forecasting with Temporal Polynomial Graph Neural Networks. In *Advances in Neural Information Processing Systems*.

[28] Zheng Lu, Chen Zhou, Jing Wu, Hao Jiang, and Songyue Cui. 2016. Integrating granger causality and vector auto-regression for traffic prediction of large-scale WLANs. *KSII Transactions on Internet and Information Systems (TIIS)* 10, 1 (2016), 136–151.

[29] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *arXiv preprint arXiv:2211.14730* (2022).

[30] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).

[31] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. *Advances in neural information processing systems* 20 (2007).

[32] Zezhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu. 2022. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4454–4458.

[33] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1567–1577.

[34] Zezhi Shao, Zhao Zhang, Wei Wei, Fei Wang, Yongjun Xu, Xin Cao, and Christian S Jensen. 2022. Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2733–2746.

[35] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. 2022. TranAD: deep transformer networks for anomaly detection in multivariate time series data. *Proceedings of the VLDB Endowment* 15, 6 (2022), 1201–1214.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[37] Leye Wang, Di Chai, Xuanzhe Liu, Liyue Chen, and Kai Chen. 2021. Exploring the generalizability of spatio-temporal traffic prediction: meta-modeling and an analytic framework. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[38] Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems* 35 (2022), 27387–27401.

[39] Xinle Wu, Dalin Zhang, Chenjuan Guo, Chaoyang He, Bin Yang, and Christian S Jensen. 2021. AutoCTS: Automated correlated time series forecasting. *Proceedings of the VLDB Endowment* 15, 4 (2021), 971–983.

[40] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining*. 753–763.

[41] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121* (2019).

[42] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5668–5675.

[43] Junchen Ye, Zihan Liu, Bowen Du, Leilei Sun, Weimiao Li, Yanjie Fu, and Hui Xiong. 2022. Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2296–2306.

[44] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[45] Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).

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

[47] Yunhao Zhang and Junchi Yan. 2022. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*.

[48] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.