EvoST: Learning Evolving Graph Structures and Temporal Patterns for Continual Spatio-Temporal Traffic Forecasting

Anonymous authors
Paper under double-blind review

Abstract

Continual spatio-temporal traffic forecasting is ought to accommodate expanding traffic sensor networks and adapt to spatial and temporal distribution shifts across successive periods. Existing methods address network evolution while underexploit node-specific spatial correlations and explicit temporal patterns. In this paper, we propose EvoST, a spatio-temporal learning paradigm that jointly learns evolving graph structures and temporal patterns for continual traffic forecasting. For spatial modeling, EvoST maintains two learnable node embeddings to parameterize adaptive graph adjacencies. At each period transition, embeddings of existing nodes are inherited and augmented with randomly initialized embeddings of newly deployed traffic sensors. The optimization in the current period retains and refines previously learned relations and discovers new dependencies involving newly added nodes. For temporal modeling, a calendar embedding dictionary independent of the sensor count is inherited and updated across periods, transferring learned periodic regularities while adapting to changing temporal patterns. We instantiate this paradigm with linearized kernel graph convolution for efficient spatial aggregation and lightweight weekly time embeddings that combine discrete low-rank representations with continuous phase features. These components enable scalable modeling of spatial correlations and temporal regularities as the traffic network expands. Comprehensive experiments on three real-world continuous traffic flow datasets demonstrate the effectiveness, versatility, efficiency, and interpretability of our method. The source codes of EvoST are available at https://anonymous.4open.science/r/EvoST for review.

1 Introduction

Spatio-temporal traffic forecasting is a classic and long-standing study that aims at mining the spatial and temporal correlations existed in road networks and predicting future traffic states, such as traffic flow, speed, and density. Numerous works have been proposed using Graph Neural Network (GNN) (Yu et al., 2018; Song et al., 2020; Shao et al., 2022b; Han et al., 2024; Wu et al., 2026), Recurrent Neural Network (RNN) (Li et al., 2018; Jiang et al., 2023b; Li et al., 2023; Gao et al., 2025), and Transformer (Zheng et al., 2020; Jiang et al., 2023a; Gao et al., 2024; Fang et al., 2025). Though effective and efficient, these methods are designed under the assumption that traffic network is stationary where the topology is fixed. However, in real-world scenarios, the number of deployed traffic observation sensors is generally increasing with the city expansion and infrastructure construction. This poses challenges for traffic forecasting methods that the spatial and temporal distributions are both changed. The illustration of these distribution shifts is provided in Appendix A.1.

In order to support continuous forecasting across successive periods, such as many years, several methods have been developed. Current works can be classified into replay-based, pattern-based, and prompt-based. Replay-based (Chen et al., 2021; Wang et al., 2023a) methods train the model on newly added traffic nodes to absorb new knowledge while detect and incorporate influential existing nodes at the same time to retain old knowledge. Pattern-based (Wang et al., 2023b; Zhang et al., 2025) methods maintain representative patterns of traffic state data to manage knowledge through pattern retrieval. Prompt-based (Chen & Liang, 2025; Liu & Zhang, 2026) methods employ

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

learnable prompts to adapt to evolving distributions while freeze the backbone to retain the learned knowledge.

These methods provide constructive solutions for traffic forecasting in continuously changing scenarios. However, there still exist limitations in these methods. Firstly, most current methods are node-agnostic which means that the model is not aware of the node counts. The trained model on one period can be directly used or finetuned in other periods even though the predefined adjacencies are different. Although this design is specifically suitable to the continuous scenarios, it limits the modeling of spatial correlations among traffic sensors that are essential for accurate predictions. Moreover, this limitation also hinders the application of advanced STGNN on continuous scenarios which equips adaptive adjacency learning technique. Secondly, most current methods are lack of explicit temporal patterns modeling. Except for the evolving graph structures and the consequent spatial distribution drift, there also exists significant temporal distribution drift across different periods. It is beneficial to learn and absorb the changing temporal patterns in continuous scenarios. Finally, most current methods tend to design complex workflows or rely on extra data which limits the generalization and application in real-world scenarios.

In this paper, we propose a new paradigm of Spatio-Temporal Network tailored to Evolving scenarios (EvoST), which learns evolving graph structures and temporal patterns with the expansion of traffic road network across successive periods. We maintain two learnable node vectors for spatial dependency learning and accordingly generate adaptive adjacencies which are used for graph convolution. The trained node vectors on current period are inherited in the next period. For the newly added traffic sensors, we assign them new node vectors with random initialization. These two parts of vectors are then combined to form new vectors specific to the node counts of new period. Through the adjacency learning in new period, the correlations of old nodes are retained and finetuned. Meanwhile, the new correlations between old and new nodes, new and new nodes are gradually learned. As for the temporal patterns, we maintain a calendar embedding dictionary to store periodic regularities of traffic state data. Since the embedding dictionary is not related to the number of traffic sensors, it can be directly reloaded and finetuned across successive periods. This process provides a well initialization containing previous temporal patterns and adaption to current patterns through gradient descent. Based on this paradigm, we further provide a concrete instance. In spatial aspect, we use linearized kernel graph convolution to approximate the original GCN with adaptive adjacencies, guaranteeing both accuracy and efficiency. While in temporal aspect, we use weekly time embeddings which consist of a discrete low-rank embedding and a continuous phase embedding to model complementary week patterns.

To evaluate our proposed method, we conduct comprehensive experiments on three real-world continuous traffic flow datasets, including a well-recognized benchmark PEMS-Stream, and PEMS06-Stream dataset with extended periods of eleven years, and a large-scale PEMS12-Stream dataset. The experimental results show the effectiveness, versatility, efficiency, and interpretability of our method. The contributions of this paper are summarized as follows: ❶ We propose a paradigm of spatio-temporal network tailored to continual traffic forecasting, which learns evolving graph structures and temporal patterns in continuous scenarios. ❷ We propose a method using linearized kernel graph convolution and lightweight time embeddings to instantiate the spatio-temporal continual paradigm, which acquires both accuracy and efficiency. ❸ We conduct comprehensive experiments on three real-world continuous traffic flow datasets to verify our method.

2 Related Work

Spatio-temporal Traffic Forecasting. There are prominent regularities existed in traffic state data, mainly reflected as spatial and temporal correlations. Thus, STGNN (Li et al., 2018; Yu et al., 2018; Guo et al., 2019; Li et al., 2023) and Transformer (Zheng et al., 2020; Xu et al., 2020; Jiang et al., 2023a; Guo et al., 2023) are widely used for correlations modeling. From the perspective of representation learning, GWNet (Wu et al., 2019), AGCRN (Bai et al., 2020), MTGNN (Wu et al., 2020) employ adaptive adjacence learning to mine the relations among traffic sensors. STID (Shao et al., 2022a), STAEformer (Liu et al., 2023) use learnable node and temporal embeddings to improve the prediction performance. MegaCRN (Jiang et al., 2023b), ST-SSDL (Gao et al., 2025) apply prototype learning to achieve more functions via assigning learnable parameter vectors various implications. Recently, more methods designed for high efficiency are developed, such as BigST

(Han et al., 2024), GSNet (Kong et al., 2025), PatchSTG (Fang et al., 2025), BiST (Ma et al., 2025b), HSTMixer (Wang et al., 2025), RAGC (Wu et al., 2026), which are tailored to large-scale scenarios.

Continual Learning. In the increasingly expanding scenarios, there simultaneously exists commonalities and differences among successive years. TrafficStream (Chen et al., 2021) train the model on the newly added traffic sensors with their two hop neighbors in each year. Meanwhile, historical nodes are replayed and model weight regularization is applied to consolidate the learned knowledge. STKEC (Wang et al., 2023a) refines the influence nodes detection strategy and utilizes memory mechanism to preserve the learned knowledge. PECPM (Wang et al., 2023b) designs a bank module to store and update representative patterns which are used to manage knowledge with pattern expansion and consolidation mechanism. STRAP (Zhang et al., 2025) constructs a trainable pattern library enriched with historical, structural, and semantic information which will be retrieved for the input data at inference stage. EAC (Chen & Liang, 2025) employs prompt parameter pool to adapt to the expansion of traffic network with frozen backbone to retain learned knowledge from previous period. STBP (Liu & Zhang, 2026) further utilizes contextual pattern bank to generate structured multi-components for the gating and attention mechanisms.

3 PRELIMINARIES

Definition 1 (Stream Traffic State Data). Suppose that traffic state data can be divided into P periods. The stream traffic state data can be formulated as $\mathcal{S} = (\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \ldots, \mathcal{D}^{(P)})$ where $\mathcal{D}^{(p)} = (\mathcal{G}^{(p)}, \mathcal{X}^{(p)})$ is the pth period of traffic network and traffic state data. $\mathcal{G}^{(p)} = (\mathcal{V}^{(p)}, \mathcal{E}^{(p)})$ denote the traffic network in period p, where $\mathcal{V}^{(p)}$ and $\mathcal{E}^{(p)}$ are the node and edge sets, respectively, and $N_p = |\mathcal{V}^{(p)}|$ is the number of traffic sensors in this period. $\mathcal{X}^{(p)} = (\mathbf{x}_t^{(p)})_{t=1}^{T_p} \in \mathbb{R}^{T_p \times N_p \times C}$ is the traffic data of pth period where T_p is the number of time steps, C is the number of input features.

Definition 2 (Spatio-Temporal Continual Traffic Forecasting). Given the stream traffic state data $\mathcal{S}$, spatio-temporal continual traffic forecasting aims to update the forecasting model continuously using the training data from current period and the preserved knowledge from previous periods. This process can be formulated as follows:

$$\begin{aligned} \boldsymbol{\theta}^{(p)} &= \mathcal{U}(\boldsymbol{\theta}^{(p-1)}, \mathcal{D}_{\text{train}}^{(p)}, \mathcal{M}^{(p-1)}), \\ \widehat{\mathbf{Y}}_t^{(p)} &= f_{\boldsymbol{\theta}^{(p)}}(\mathbf{X}_t^{(p)}, \mathcal{G}^{(p)}), \end{aligned} \tag{1}$$

where $\mathcal{U}$ is the continual learning procedure, $\mathcal{D}_{\text{train}}^{(p)}$ refers to the training samples from period p, and $\mathcal{M}^{(p-1)}$ is an optional item denoting the retained knowledge from previous periods. $\mathbf{X}_t^{(p)} = \mathcal{X}_{t-L+1:t}^{(p)} \in \mathbb{R}^{L \times N_p \times C}$ are the L time steps historical observations and $\mathbf{Y}_t^{(p)} = \mathcal{X}_{t+1:t+H}^{(p)} \in \mathbb{R}^{H \times N_p \times C}$ are H time steps future traffic state. $\widehat{\mathbf{Y}}_t^{(p)} \in \mathbb{R}^{H \times N_p \times C}$ denotes the predictions.

4 METHODOLOGY

In this section, we firstly describe the paradigm of EvoST and then provide a concrete instance. EvoST is a Spatio-Temporal Network tailored to evolving scenarios of traffic forecasting. To adapt to the evolving graph structures, learnable and extensible node embeddings are maintained. When the data of new period comes, the node embeddings of existing traffic sensors are inherited from previous period while the node embeddings of new sensors are randomly initialized. The concatenated node embeddings are then used to learn the adaptive adjacencies through which the relations of existing nodes are preserved and updated while the relations of new nodes are learned. To adapt to the evolving temporal patterns, a calendar embedding dictionary is maintained and finetuned across successive periods. The proposed instance of EvoST contains a linearized kernel graph convolution and weekly time embeddings consist of a discrete low-rank embedding and a continuous phase embedding, which learns the evolving spatial and temporal distributions while guarantees efficiency.

4.1 PARADIGM OF EVOST

As shown in Figure 1, EvoST maintains learnable node embeddings that will be expanded with periods progress. Thus, the shape of model parameters of current period is different from previous

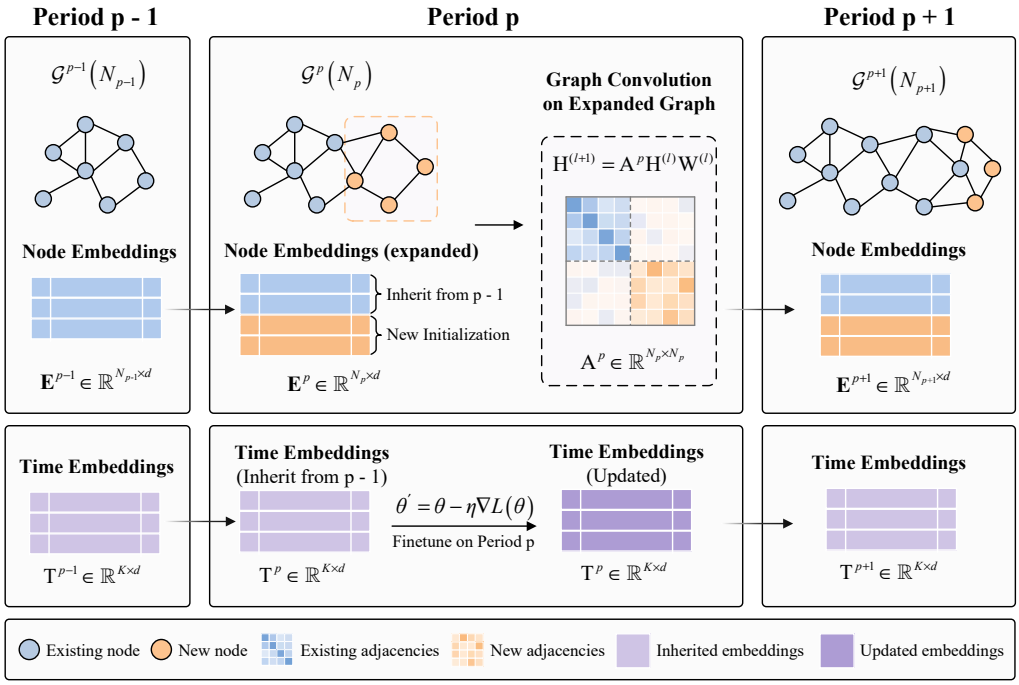

Figure 1: The paradigm of EvoST.

periods. We define the model parameters of current period p as four parts:

$$\mathbf{\Theta}^{(p)} = (\mathbf{\Theta}^{(p)}_{\text{shared}}, \mathbf{E}^{(p)}_{\text{time}}, \mathbf{E}^{(p)}_{1}, \mathbf{E}^{(p)}_{2}), \tag{2}$$

where $\mathbf{E}^{(p)}_{1}, \mathbf{E}^{(p)}_{2} \in \mathbb{R}^{N_p \times d_n}$ are two learnable node embeddings and $\mathbf{E}^{(p)}_{\text{time}} \in \mathbb{R}^{K \times d_t}$ is the calendar embedding dictionary. $\mathbf{\Theta}_{\text{shared}}$ denotes all other model parameters. N_p and K refer to the number of nodes of period p and the number of time slots, respectively. d_n and d_t are embedding dimensions. The initialization of model parameters in current period can be formulated as follows:

$$\mathbf{\Theta}^{(p,0)}_{\text{shared}} \leftarrow \widehat{\mathbf{\Theta}}^{(p-1)}_{\text{shared}}, \quad \mathbf{E}^{(p,0)}_{\text{time}} \leftarrow \widehat{\mathbf{E}}^{(p-1)}_{\text{time}}, \tag{3}$$

$$\mathbf{E}^{(p,0)}_{1} = \widehat{\mathbf{E}}^{(p-1)}_{1} \,\|\, \mathbf{E}^{(p,0)}_{1,\text{new}}, \quad \mathbf{E}^{(p,0)}_{2} = \widehat{\mathbf{E}}^{(p-1)}_{2} \,\|\, \mathbf{E}^{(p,0)}_{2,\text{new}}, \tag{4}$$

where $\mathbf{E}^{(p,0)}_{1,\text{new}}, \mathbf{E}^{(p,0)}_{2,\text{new}} \in \mathbb{R}^{(N_p - N_{p-1}) \times d_n}$ refer to newly added parameters which are initialized randomly. The superscript $(p, 0)$ means the initialized state of period p. The symbol $\leftarrow$ denotes parameter assignment and $\|$ is the concatenation operation. Having the expanded node embeddings, we can derive the adaptive adjacencies using generator functions (Wu et al., 2019; 2020; Bai et al., 2020; Ma et al., 2025a). This process can be described as follows:

$$\mathbf{A}^{(p,0)} = \mathcal{G}(\mathbf{E}^{(p,0)}_{1}, \mathbf{E}^{(p,0)}_{2}), \quad \mathcal{G} : \mathbb{R}^{N \times d_n} \times \mathbb{R}^{N \times d_n} \to \mathbb{R}^{N \times N} \tag{5}$$

We then train the model on current data $\mathcal{D}_p$ and achieve the optimized parameters $\widehat{\mathbf{\Theta}}^{(p)}$, which will be used for inference at current period p and also used for initialization at next period $p + 1$.

$$\widehat{\mathbf{\Theta}}^{(p)} = \text{Train}(\mathcal{D}_p; \mathbf{\Theta}^{(p,0)}_{\text{shared}}, \mathbf{E}^{(p,0)}_{\text{time}}, \mathbf{A}^{(p,0)}). \tag{6}$$

4.2 Weekly Time Embeddings

In following formulations, we omit the superscript of p. The subscript of t is the end time step of input data window. Suppose that each day contains T_d time slots and each week contains D_w days, we can know that one week has $T_w = T_d D_w$ time slots. The slot index of week is calculated as:

$$k_t = T_d w_t + s_t, \quad k_t \in \{0, \ldots, T_w - 1\}, \tag{7}$$

where s_t and w_t denote the slot index of day and week index, respectively. To avoid directly maintaining a high dimensional embedding dictionary, we build a low-rank factor dictionary firstly and then project the retrieved embeddings into high dimensional space.

$$\mathbf{e}_{\text{disc},t} = \mathbf{F}_{\text{week}}\left[k_t, :\right]\mathbf{W}_{\text{factor}} + \mathbf{b}_{\text{factor}}, \tag{8}$$

where $\mathbf{F}_{\text{week}} \in \mathbb{R}^{T_w \times r}$ is the factor dictionary and r is the intermediate dimension. $\mathbf{W}_{\text{factor}} \in \mathbb{R}^{r \times d_t}$ and $\mathbf{b}_{\text{factor}} \in \mathbb{R}^{1 \times d_t}$ are weight and bias matrices of this projection, respectively. $\mathbf{e}_{\text{disc},t} \in \mathbb{R}^{1 \times d_t}$ is the obtained discrete weekly time embedding of time step t. The efficiency analysis of low-rank embedding is provided in Appendix A.2.3. Besides, we further incorporate explicit weekly phase features and then transform the features into high dimensional space with linear projection:

$$\mathbf{c}(k_t) = \left[\sin\left(2\pi k_t/T_w\right), \cos\left(2\pi k_t/T_w\right)\right], \quad \mathbf{e}_{\text{phase},t} = \mathbf{c}(k_t)\mathbf{W}_{\text{phase}}, \tag{9}$$

where $\mathbf{W}_{\text{phase}} \in \mathbb{R}^{2 \times d_t}$ is the weight matrix and $\mathbf{e}_{\text{phase},t} \in \mathbb{R}^{1 \times d_t}$ is the obtained continuous phase embedding of time step t. The discrete low-rank embedding provides time slot related learnable representations and continuous phase embedding provides explicit periodic features. The final weekly time embeddings are the addition of these two components that are then broadcasted to all nodes and concatenated with the input data embedding along the feature dimension.

$$\mathbf{e}_{\text{time},t} = \mathbf{e}_{\text{disc},t} + \mathbf{e}_{\text{phase},t}, \quad \mathbf{H}_t = \mathbf{H}_{\text{input},t} \parallel (\mathbf{1}_N \mathbf{e}_{\text{time},t}), \tag{10}$$

where $\mathbf{1}_N \in \mathbb{R}^{N \times 1}$ is the vector containing value 1. $\mathbf{H}_{\text{input},t} \in \mathbb{R}^{N \times d_x}$ is the input data embedding and $\mathbf{H}_t \in \mathbb{R}^{N \times (d_x + d_t)}$ is the fused embedding fed for subsequent modules.

4.3 Linearized Kernel Graph Convolution

In following formulations, we omit the superscript of p, dimensions of batch size and time steps, and set $N = N_p$. Suppose that the input embeddings of graph convolution is $\mathbf{H} \in \mathbb{R}^{N \times D}$ and the two learnable node embeddings are $\mathbf{E}_{\text{recv}}, \mathbf{E}_{\text{send}} \in \mathbb{R}^{N \times d_n}$.

We use exponential kernel with constant basic item to describe the potential correlations among nodes where $\alpha \geq 0$ control the basic correlation strength shared in all node pairs. When $\alpha = 1$, the row-normalized exponential kernel is exactly equivalent to a Softplus-based adaptive adjacency. Since Softplus is a smooth approximation to ReLU, the kernel provides a smooth surrogate for the commonly used ReLU-based adaptive adjacency. When $\alpha = 0$, the kernel reduces exactly to the standard dot-product softmax adjacency. We provide more details in Appendix A.2.1.

$$\kappa(\mathbf{u}, \mathbf{v}) = \alpha + \exp(\mathbf{u}^\top \mathbf{v}). \tag{11}$$

In order to avoid explicitly computing kernel scores of all node pairs, we use positive random features (Choromanski et al., 2020) to approximate kernel scores as follows. The details of kernel features ϕ and ψ are provided in Appendix A.2.2.

$$\widehat{\kappa}(\mathbf{u}, \mathbf{v}) = \boldsymbol{\psi}(\mathbf{u})^\top \boldsymbol{\psi}(\mathbf{v}) = \alpha + \sum_{a=1}^{m} \phi_a(\mathbf{u})\phi_a(\mathbf{v}) \approx \kappa(\mathbf{u}, \mathbf{v}). \tag{12}$$

We then map the two node embeddings into kernel feature space:

$$\mathbf{Q} = \boldsymbol{\psi}(\mathbf{E}_{\text{recv}}), \quad \mathbf{K} = \boldsymbol{\psi}(\mathbf{E}_{\text{send}}), \tag{13}$$

where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{N \times (m+1)}$. Based on inner product of kernel features and the row normalization, we can define the adaptive adjacency matrix as follows. Kindly notice that $\mathbf{A} \in \mathbb{R}^{N \times N}$ is merely used for defining the propagation operator and is not explicitly built in implementation.

$$\mathbf{A} = \boldsymbol{\Delta}^{-1}\mathbf{Q}\mathbf{K}^\top, \quad \boldsymbol{\Delta} = \operatorname{diag}\left(\mathbf{Q}\left(\mathbf{K}^\top \mathbf{1}_N\right)\right). \tag{14}$$

Therefore, graph convolution can be formulated applying associative property of matrix multiplication as follows. Besides, this process can also be seen as aggregating all node features into fixed-dimension kernel feature space firstly and then assigning back to all nodes according to the normalized matrix.

$$\mathcal{P}(\mathbf{H}) = \mathbf{A}\mathbf{H} = \overline{\mathbf{Q}}\left(\mathbf{K}^\top \mathbf{H}\right), \quad \overline{\mathbf{Q}} = \boldsymbol{\Delta}^{-1}\mathbf{Q}. \tag{15}$$

Finally, we can formulate the linearized kernel graph convolution with multiple diffusion steps as follows where $\mathbf{W}_g^{(\ell)} \in \mathbb{R}^{D \times D}$ is the weight matrix of the ℓ step and $\mathbf{b}_g \in \mathbb{R}^D$ is the bias matrix.

$$\mathcal{G}(\mathbf{H}) = \sum_{\ell=0}^{S} \mathbf{H}^{(\ell)}\mathbf{W}_g^{(\ell)} + \mathbf{1}_N \mathbf{b}_g^\top, \quad \mathbf{H}^{(\ell)} = \overline{\mathbf{Q}}\left(\mathbf{K}^\top \mathbf{H}^{(\ell-1)}\right), \ell = 1, \ldots, S. \tag{16}$$

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets. We conduct experiments on three real-world continuous traffic flow datasets, including PEMS-Stream, PEMS06-Stream and PEMS12-Stream. PEMS-Stream (Chen et al., 2021) is a widely recognized benchmark used in continual spatio-temporal traffic forecasting which spans seven years. PEMS06-Stream and PEMS12-Stream are initially introduced in EvoXXLTraffic (Yin et al., 2026) from which we select successive eleven and four years data, respectively, and extract three months data for each year. Moreover, PEMS12-Stream is also characterized as a large-scale dataset compared to others. For each dataset, we divide and obtain the training, validation, and test sets with the ratio of 6:2:2. More details about these datasets are provided in Appendix A.3.1.

Baselines and Evaluation Metrics. We compare our method to eight baselines: Retrain-ST, Pretrain-ST, Online-ST-AN, Online-ST-NN, TrafficStream, STKEC, EAC, STBP. Specifically, Retrain-ST, Pretrain-ST, Online-ST-AN, Online-ST-NN are four naive methods that based on the same basic STGNN (Chen et al., 2021) while have different training strategies. TrafficStream (Chen et al., 2021) and STKEC (Wang et al., 2023a) trains models on newly added traffic nodes and the detected influential nodes or replayed nodes. EAC (Chen & Liang, 2025) and STBP (Liu & Zhang, 2026) freeze the backbone to retain the learned knowledge and utilize learnable prompts or pattern bank to adapt to the evolving distributions. We adopt three commonly used metrics MAE, RMSE, MAPE to evaluate the baselines and our method. More details about these baselines and evaluation metrics are provided in Appendix A.3.2 and A.3.3.

Implementation Details. The time dimension is set to 128, 64, 64 on PEMS-Stream, PEMS06-Stream, PEMS12-Stream, respectively. The time embedding rank, node dimension, kernel constant, diffusion step are set to 8, 16, 1, 2, respectively, for all datasets. All experiments are conducted using an uniform framework with five runs on NVIDIA A100 40GB GPU. The average results are reported. More details about the implementation are provided in Appendix A.3.4.

5.2 PERFORMANCE COMPARISONS

As shown in Table 1, EvoST achieves the optimal results on all datasets, which indicates the effectiveness of our method. Moreover, we have the following observations. The model Pretrain-ST performs worse than Retrain-ST, which indicates that new knowledge should be absorbed in continuous scenarios. Meanwhile, Online-ST-NN underperforms Online-ST-AN reflecting the necessity of retaining old knowledge. It is also the phenomenon of catastrophic forgetting that training merely on new nodes will lose the capability of accurately predicting on old nodes. TrafficStream and STKEC are also trained on new nodes since it is more efficient than training on all nodes. However, they further incorporate the replayed or detected influential old nodes to retain old knowledge, thus having better performance than Online-ST-NN. EAC and STBP introduce the learnable prompts and pattern bank to adapt to the evolving distributions, having a large performance improvement compared to former baselines. However, the strategy of freezing backbone will incur instability in some cases, such as the large RMSE and MAPE metrics which are sensitive to large and small values, respectively. We unlock the backbone parameters of STBP on PEMS06-Stream and PEMS12-Stream and report the results since the original mode will have exceptional errors. In addition to our proposed EvoST, we also propose a method Ada-ST, which has the same STGNN backbone with the four naive models, TrafficStream, STKEC, and EAC. Besides, it equips with the spatial modeling paradigm of EvoST and adopts the most basic adaptive adjacency generator (Wu et al., 2019). The results show that Ada-ST performs better than previous methods at most metrics on all datasets, which also reflects the effectiveness of evolving graph structure learning in EvoST.

Table 2: Performance comparison of EAC and EAC with evolving graph structure learning.

Datasets	Metric	EAC	EAC+	Δ
PEMS-Stream	MAE	14.61 ± 0.04	12.93 ± 0.09	-11.50%
	RMSE	24.15 ± 0.09	21.55 ± 0.16	-10.77%
	MAPE(%)	19.54 ± 0.15	17.36 ± 0.17	-11.16%
PEMS06-Stream	MAE	13.64 ± 0.13	12.77 ± 0.09	-6.38%
	RMSE	22.61 ± 0.48	21.30 ± 0.53	-5.79%
	MAPE(%)	23.25 ± 1.65	23.25 ± 1.75	-0.00%
PEMS12-Stream	MAE	14.44 ± 0.07	13.91 ± 0.47	-3.67%
	RMSE	25.97 ± 0.22	25.23 ± 0.84	-2.85%
	MAPE(%)	31.42 ± 2.23	31.30 ± 1.47	-0.38%

We also investigate the versatility of our evolving graph structure learning module. We embed it into the baseline EAC and replace the original static predefined graph adjacencies, forming the variant

Table 1: Performance comparison on three continuous traffic flow datasets. Best results are highlighted in **red bold**, and second-best results are marked with blue underlines.

Datasets		*PEMS-Stream* (655 → ··· → 871)				*PEMS06-Stream* (418 → ··· → 746)				*PEMS12-Stream* (2365 → ··· → 2538)			
Method	**Metric**	**3**	**6**	**12**	**Avg.**	**3**	**6**	**12**	**Avg.**	**3**	**6**	**12**	**Avg.**
Retrain-ST	MAE	13.18±0.01	15.25±0.01	20.07±0.02	15.68±0.01	11.99±0.04	13.38±0.03	16.12±0.02	13.56±0.02	13.09±0.04	15.23±0.07	19.49±0.09	15.51±0.06
	RMSE	21.62±0.02	25.20±0.02	32.79±0.04	26.11±0.02	19.51±0.06	22.02±0.06	26.56±0.04	22.39±0.04	23.38±0.07	27.39±0.14	34.98±0.17	28.18±0.12
	MAPE(%)	17.24±0.18	19.66±0.30	25.12±0.53	20.17±0.22	17.47±0.24	19.32±0.15	22.80±0.16	19.52±0.14	22.50±1.04	25.13±0.62	30.96±0.49	25.65±0.59
Pretrain-ST	MAE	13.28±0.02	15.46±0.03	20.42±0.06	15.89±0.04	12.24±0.05	13.80±0.06	16.91±0.03	14.01±0.02	13.18±0.12	15.37±0.10	19.69±0.09	15.66±0.08
	RMSE	21.79±0.05	25.62±0.07	33.55±0.12	26.55±0.07	19.86±0.08	22.64±0.11	27.81±0.27	23.13±0.11	23.39±0.18	27.43±0.16	35.10±0.17	28.25±0.14
	MAPE(%)	17.77±0.58	20.35±0.71	26.42±1.09	21.03±0.71	21.69±2.39	23.85±2.76	27.93±2.46	24.05±2.60	29.53±4.29	31.97±4.36	39.04±3.22	32.63±3.85
Online-ST-AN	MAE	13.10±0.01	15.13±0.03	19.86±0.04	15.56±0.02	11.88±0.03	13.25±0.03	15.92±0.03	13.42±0.02	13.02±0.04	15.15±0.05	19.36±0.07	15.42±0.05
	RMSE	21.46±0.03	24.98±0.05	32.38±0.09	25.85±0.04	19.30±0.05	21.78±0.09	26.17±0.10	22.13±0.07	23.27±0.07	27.22±0.07	34.71±0.10	28.00±0.07
	MAPE(%)	17.23±0.38	19.54±0.27	24.97±0.45	20.05±0.30	17.59±0.27	19.23±0.19	22.80±0.28	19.53±0.16	22.76±0.69	25.56±0.96	31.56±0.88	26.11±0.78
Online-ST-NN	MAE	22.51±7.12	27.03±7.22	39.31±9.84	28.73±7.37	12.61±0.08	14.35±0.08	17.61±0.10	14.55±0.08	13.65±0.11	16.17±0.13	21.09±0.38	16.49±0.18
	RMSE	47.61±20.36	55.69±19.68	83.43±26.81	62.84±20.70	20.86±0.93	24.13±0.90	29.81±0.99	24.67±0.93	24.43±0.22	29.22±0.25	38.12±0.70	30.19±0.40
	MAPE(%)	24.44±4.37	28.64±5.18	39.33±6.94	29.97±5.11	22.17±1.80	24.32±1.75	29.05±1.44	24.75±1.73	24.39±1.25	27.12±1.43	34.36±0.88	27.95±0.68
TrafficStream	MAE	13.42±0.04	15.70±0.06	20.82±0.15	16.15±0.08	12.12±0.06	13.60±0.04	16.42±0.04	13.77±0.04	13.09±0.06	15.25±0.03	19.56±0.07	15.54±0.04
	RMSE	22.13±0.20	26.11±0.19	34.43±0.50	27.17±0.30	19.86±0.34	22.56±0.35	27.29±0.45	22.94±0.37	23.38±0.11	27.44±0.06	35.10±0.14	28.23±0.09
	MAPE(%)	17.41±0.20	19.90±0.35	25.96±0.61	20.58±0.39	18.47±0.47	20.20±0.39	24.03±0.59	20.58±0.43	23.53±1.36	25.48±0.56	31.58±0.45	26.33±0.72
STKEC	MAE	13.48±0.05	15.73±0.05	20.88±0.11	16.19±0.07	12.15±0.07	13.58±0.04	16.43±0.08	13.76±0.05	13.06±0.03	15.20±0.04	19.51±0.09	15.50±0.03
	RMSE	22.27±0.26	26.26±0.24	34.90±1.03	27.40±0.51	19.80±0.19	22.56±0.41	27.22±0.25	22.86±0.18	23.30±0.09	27.28±0.06	34.98±0.16	28.13±0.06
	MAPE(%)	17.67±0.29	20.02±0.22	25.86±0.40	20.66±0.22	18.61±0.52	20.21±0.46	24.21±0.62	20.65±0.47	23.40±0.71	26.05±0.96	32.63±1.09	26.87±0.63
EAC	MAE	12.94±0.03	14.41±0.07	17.57±0.08	14.61±0.04	12.20±0.11	13.52±0.14	15.75±0.25	13.64±0.13	12.92±0.06	14.33±0.06	16.97±0.15	14.44±0.07
	RMSE	21.14±0.04	23.75±0.11	28.82±0.18	24.15±0.09	19.84±0.54	22.25±0.47	26.14±0.57	22.61±0.48	22.85±0.16	25.64±0.19	30.53±0.39	25.97±0.22
	MAPE(%)	17.34±0.63	18.65±0.05	23.39±1.25	19.54±0.15	22.27±1.52	23.85±1.81	24.57±1.77	23.25±1.65	29.52±1.30	31.32±2.96	34.32±5.16	31.42±2.23
STBP	MAE	11.48±0.09	12.24±0.14	13.72±0.26	12.30±0.15	11.73±0.36	12.49±0.39	14.10±0.42	12.60±0.33	12.52±0.10	13.68±0.16	15.97±0.38	13.77±0.18
	RMSE	19.07±0.14	20.55±0.23	23.11±0.51	20.66±0.28	29.78±12.55	33.71±18.40	42.74±19.86	36.35±19.05	25.11±1.76	30.48±5.36	39.97±9.69	31.40±5.81
	MAPE(%)	14.79±0.22	15.73±0.29	17.43±0.45	15.75±0.13	23.51±2.86	25.89±4.75	31.43±5.35	26.40±3.57	23.10±0.86	25.29±0.89	29.78±1.77	25.60±0.80
Ada-ST	MAE	11.97±0.04	12.98±0.08	14.78±0.11	13.02±0.07	11.85±0.13	12.84±0.14	14.68±0.19	12.94±0.15	12.71±0.14	14.03±0.13	16.55±0.16	14.14±0.12
	RMSE	19.66±0.02	21.46±0.08	24.39±0.12	21.55±0.07	19.28±0.19	21.20±0.28	24.39±0.62	21.37±0.31	22.55±0.21	25.16±0.17	29.90±0.23	25.50±0.19
	MAPE(%)	17.06±0.80	17.70±0.63	20.04±0.51	17.95±0.61	18.27±0.46	19.45±0.42	22.16±0.70	19.66±0.46	23.89±1.76	25.36±1.80	30.41±2.76	26.03±1.94
EvoST	MAE	**11.12**±0.03	**11.83**±0.05	**12.95**±0.07	**11.83**±0.05	**11.02**±0.03	**11.81**±0.05	**13.01**±0.10	**11.81**±0.06	**11.83**±0.04	**12.89**±0.08	**14.52**±0.15	**12.89**±0.09
	RMSE	**18.60**±0.06	**19.96**±0.09	**21.78**±0.10	**19.89**±0.08	**17.99**±0.03	**19.46**±0.07	**21.47**±0.14	**19.44**±0.08	**21.44**±0.38	**23.89**±1.15	**27.70**±2.79	**24.08**±1.48
	MAPE(%)	**14.67**±0.13	**15.60**±0.17	**17.39**±0.21	**15.69**±0.14	**16.49**±0.31	**17.67**±0.36	**19.73**±0.54	**17.74**±0.37	**21.79**±0.23	**23.59**±0.13	**26.63**±0.46	**23.69**±0.22

EAC+. Thus, the evolving spatial relations between nodes are learned and utilized for graph convolution. As shown in Table 2, EAC+ performs better than EAC on all datasets, with up to 11.50% reductions of MAE on PEMS-Stream dataset. This reflects that the evolving graph structure learning module has great potentials to replace predefined adjacencies in continuous traffic forecasting scenarios that the nodes are expanding within successive periods.

5.3 Ablation Study and Hyperparameter Sensitivity Analysis

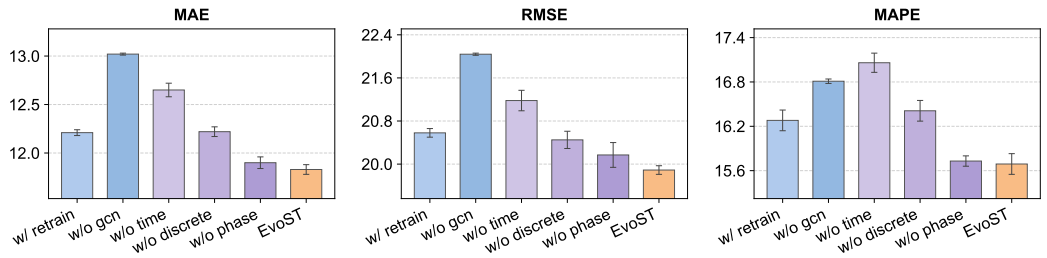

Figure 2: Ablation study on PEMS-Stream.

We design five variants to evaluate the compoents of EvoST. The variant w/ retrain trains models from scratch in each period instead of inheriting model weights from previous periods. w/o gcn removes the linearized kernel graph convolution module. w/o time removes weekly time embeddings. w/o discrete removes the discrete low-rank embedding while w/o phase removes the continuous phase embedding. The ablation results on PEMS-Stream are shown in Figure 2. It can be seen that the variant w/ retrain performs worse than EvoST, indicating the effectiveness of knowledge transferring between successive periods. As the main encoding module, the linearized kernel graph convolution plays an important role in spatial correlations modeling. Removing it will incur large performance degradation, as reflected in the variant w/o gcn. Similarly, the variant w/o time under-

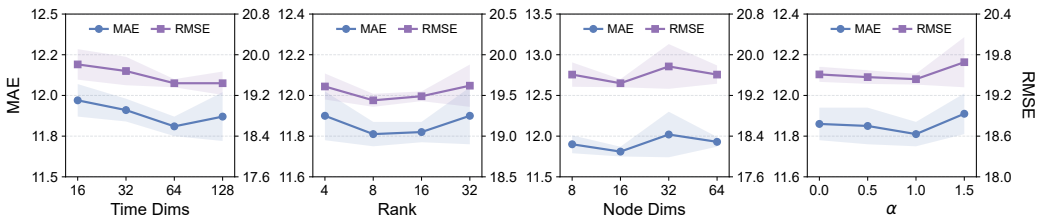

Figure 3: Analysis of four hyperparameters on PEMS06-Stream.

performs EvoST, which indicates the essential of time embeddings in EvoST that models temporal regularities while adapting to evolving distributions. Furthermore, we investigate the functions of discrete low-rank embedding and continuous phase embedding, which are ingredients of the time embeddings. It can be seen that the variants w/o discrete and w/o phase perform better than w/o time and meanwhile perform worse than EvoST. This comparison reflects the effectiveness of these two embeddings. We also conduct ablations on PEMS06-Stream that are provided in Appendix A.4.2.

We consider the sensitivity of four hyperparameters in EvoST. The analysis results on PEMS06-Stream dataset are shown in Figure 3. The optimal values of time embedding dimension and node embedding dimension are 64 and 16, respectively, after which the model will incur larger errors and instabilities. The rank of 8 enables the optimal results which also reflects the high efficiency of the discrete low-rank time embedding. The constant item α in kernel features achieves the optimal at the value of 1.0. We also conduct analysis on PEMS-Stream that are provided in Appendix A.4.3.

5.4 Efficiency Study

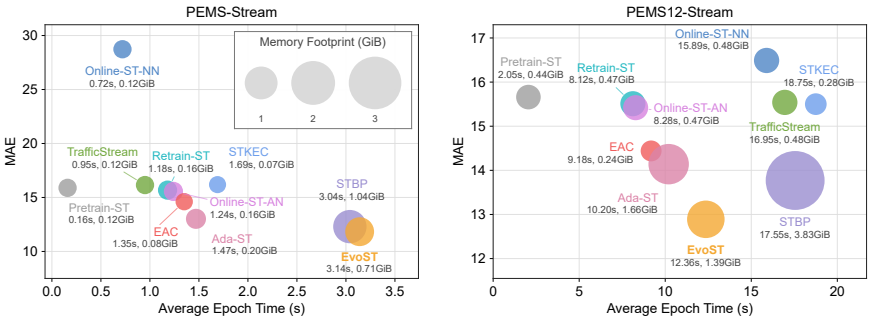

Figure 4: The efficiency study on PEMS-Stream and PEMS12-Stream datasets.

We report the average training epoch time and GPU memory of baselines and EvoST. To reflect the tradeoff between efficiency and performance, we also include the MAE metric. As shown in Figure 4, EvoST and STBP have lower prediction errors compared to other methods while requiring more training time and GPU memory. The another proposed method in this paper, Ada-ST, consumes little more time and GPU memory and has performance improvement compared to the baselines with the same backbone. On larger dataset PEMS12-Stream, the increments of required time and GPU memory in EvoST are less than most baselines which reflects its high applicability in large-scale scenarios. Specifically, the used GPU memory of STBP is increased much faster than EvoST. Besides, the training epoch time of Online-ST-NN, TrafficStream, and STKEC are also much larger. This may be caused by the process of subgraph construction and influential nodes detection that are more time-consuming on larger graphs. We also notice that EvoST consumes less GPU memory than Ada-ST, which is attributed to the linearized kernel graph convolution that does not explicitly build fully-connected adjacencies. Thus, EvoST has linear complexity with regard to the node counts.

5.5 Visualization

We firstly consider the evolving graph structures learned in EvoST. We calculate the differences of learned adjacencies between successive periods via the subtraction operation. As shown in the first

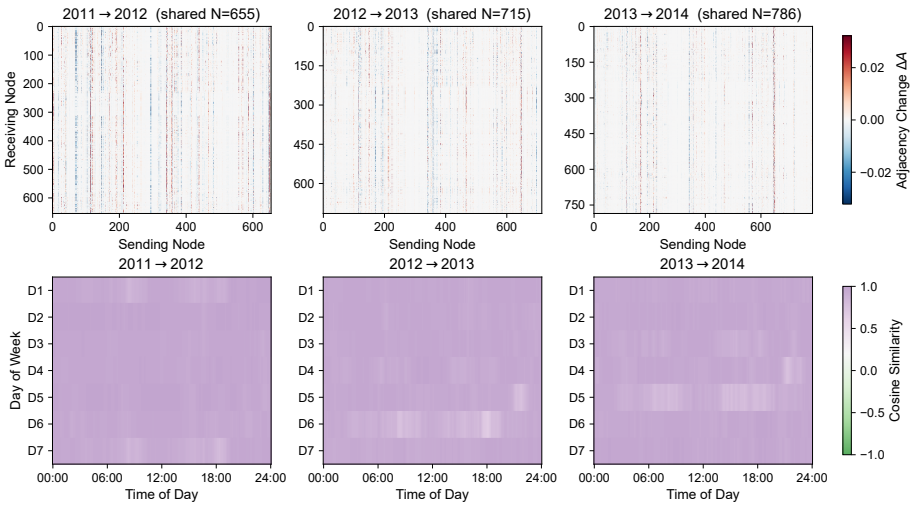

Figure 5: Evolving graph structures and temporal patterns across successive periods.

row of Figure 5, the adjacencies of most nodes are kept unchanged while several nodes have larger or smaller sending weights. This reflects that the learned adjacencies in EvoST are stable while capable of adapting to the changing spatial relations. We also investigate the learned adjacencies involving newly added nodes which is provided in Appendix A.5.2. We then consider the temporal patterns learned in EvoST across periods, as shown in the second row of Figure 5. In each period, we obtain a weekly time embeddings that contain the discrete low-rank embedding and continuous phase embedding. The cosine similarities of time embeddings between successive periods are computed. Similar to the graph structures, the learned time embeddings are stable across periods and also accommodate the evolving temporal patterns. We also analyze the learned spatial and temporal evolutions in other periods of PEMS-Stream that is provided in Appendix A.5.1.

We conduct principal component analysis on weekly time embeddings and draw the trajectories of time slots in the shared PCA space. The adjacent positions in the trajectory represent the embeddings of successive time slots of one week. From Figure 6, we can know that the embedding trajectory of each period exhibit pronounced periodic patterns with irregular local fluctuations, which are driven by the synergy of continuous phase embedding and discrete low-rank embedding. We also visualize these two components that are provided in Appendix A.5.3. Besides, the learned time embeddings also reflect significant evolving patterns across successive periods indicating its adaptability to the changing temporal distributions. We further project the embeddings into the primary direction (PC1) and draw the weekly temporal patterns that are provided in Appendix A.5.3.

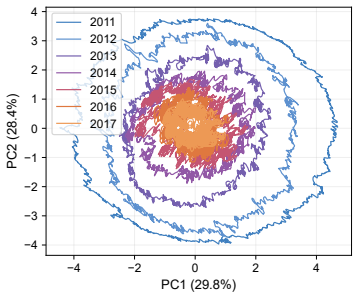

Figure 6: PCA of time embeddings across different periods.

6 Conclusion

In this paper, we propose EvoST, a spatio-temporal learning paradigm for continual traffic forecasting. We utilize the expansible learnable parameter vectors to parametrize graph structures. At the transition of successive periods, the relations among old nodes are retained and refined while the new relations involving newly added nodes are gradually learned. We also employ calendar embedding dictionary to learn the periodic regularities and adapt to changing temporal patterns, which are inherited and updated across periods. Based on this paradigm, we provide a specific instance that utilizes linearized kernel graph convolution and weekly time embeddings consist of discrete low-rank embedding and continuous phase embedding for scalable spatial and temporal modeling. We conduct comprehensive experiments on three real-world continuous traffic flow datasets. The results show the effectiveness, versatility, efficiency, and interpretability of our method.

AI USE STATEMENT

In this work, we used generative AI tools for assist with translation. We have not used generative AI tools for formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, support qualitative and thematic data analysis, interpret results, and Generate synthetic data sets, help develop theoretical models or conceptual frameworks, clean and reformat dataset are not applicable to this work. Additionally, we used generative AI tools for edit a research paper to improve readability. We have reviewed all AI-assisted work. We checked the texts translated or polished by generative AI for correctness, coherence, and readability. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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

Wei Chen and Yuxuan Liang. Expand and compress: Exploring tuning principles for continual spatio-temporal graph forecasting. In *International Conference on Learning Representations*, volume 2025, pp. 81631–81656, 2025.

Xu Chen, Junshan Wang, and Kunqing Xie. Trafficstream: A streaming traffic flow forecasting framework based on graph neural networks and continual learning. *arXiv preprint arXiv:2106.06273*, 2021.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Yuchen Fang, Yuxuan Liang, Bo Hui, Zezhi Shao, Liwei Deng, Xu Liu, Xinke Jiang, and Kai Zheng. Efficient large-scale traffic forecasting with transformers: A spatial data management perspective. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 307–317, 2025.

Haotian Gao, Renhe Jiang, Zheng Dong, Jinliang Deng, Yuxin Ma, and Xuan Song. Spatial-temporal-decoupled masked pre-training for spatiotemporal forecasting. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 3998–4006, 2024.

Haotian Gao, Zheng Dong, Jiawei Yong, Shintaro Fukushima, Kenjiro Taura, and Renhe Jiang. How different from the past? spatio-temporal time series forecasting with self-supervised deviation learning. *Advances in Neural Information Processing Systems*, 38:146647–146675, 2025.

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

Shengnan Guo, Youfang Lin, Letian Gong, Chenyu Wang, Zeyu Zhou, Zekai Shen, Yiheng Huang, and Huaiyu Wan. Self-supervised spatial-temporal bottleneck attentive network for efficient long-term traffic forecasting. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 1585–1596. IEEE, 2023.

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. *Proceedings of the VLDB Endowment*, 17(5):1081–1090, 2024.

Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 4365–4373, 2023a.

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

Weiyang Kong, Kaiqi Wu, Sen Zhang, and Yubao Liu. Graphsparsenet: A novel method for large scale traffic flow prediction. *Proceedings of the VLDB Endowment*, 18(7):2295–2307, 2025.

Fuxian Li, Jie Feng, Huan Yan, Guangyin Jin, Fan Yang, Funing Sun, Depeng Jin, and Yong Li. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Transactions on Knowledge Discovery from Data*, 17(1):1–21, 2023.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

Aoyu Liu and Yaying Zhang. A general spatio-temporal backbone with scalable contextual pattern bank for urban continual forecasting. In *International Conference on Learning Representations*, volume 2026, pp. 72080–72103, 2026.

Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 4125–4129, 2023.

Jiaming Ma, Binwu Wang, Guanjun Wang, Kuo Yang, Zhengyang Zhou, Pengkun Wang, Xu Wang, and Yang Wang. Less but more: Linear adaptive graph learning empowering spatiotemporal forecasting. *Advances in Neural Information Processing Systems*, 38:58382–58418, 2025a.

Jiaming Ma, Binwu Wang, Pengkun Wang, Zhengyang Zhou, Xu Wang, and Yang Wang. Bist: A lightweight and efficient bi-directional model for spatiotemporal prediction. *Proceedings of the VLDB Endowment*, 18(6):1663–1676, 2025b.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Zezhi Shao, Zhao Zhang, Fei Wang, Wei Wei, and Yongjun Xu. 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*, pp. 4454–4458, 2022a.

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

Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu 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*, volume 34, pp. 914–921, 2020.

Binwu Wang, Yudong Zhang, Jiahao Shi, Pengkun Wang, Xu Wang, Lei Bai, and Yang Wang. Knowledge expansion and consolidation for continual traffic prediction with expanding graphs. *IEEE Transactions on Intelligent Transportation Systems*, 24(7):7190–7201, 2023a.

Binwu Wang, Yudong Zhang, Xu Wang, Pengkun Wang, Zhengyang Zhou, Lei Bai, and Yang Wang. Pattern expansion and consolidation on evolving graphs for continual traffic prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2223–2232, 2023b.

Yongyao Wang, Jingyuan Wang, Xie Yu, Jiahao Ji, and Chao Li. Hstmixer: A hierarchical mlp-mixer for large-scale traffic forecasting. *arXiv preprint arXiv:2512.07854*, 2025.

Kaiqi Wu, Weiyang Kong, Sen Zhang, Zitong Chen, and Yubao Liu. Efficient traffic forecasting on large-scale road network by regularized adaptive graph convolution. In *2026 IEEE 42th international conference on data engineering (ICDE)*, 2026.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 1907–1913, 2019.

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

Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

Du Yin, Hao Xue, Arian Prabowo, Shuang Ao, and Flora Salim. From xxltraffic to evoxxltraffic: Scaling traffic forecasting to sensor-evolving networks. *arXiv preprint arXiv:2605.29768*, 2026.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 3634–3640, 2018.

Haoyu Zhang, Hao Miao, Xinke Jiang, Yuchen Fang, and Yifan Zhang. Strap: Spatio-temporal pattern retrieval for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 38:118006–118041, 2025.

Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 1234–1241, 2020.

A Appendix

A.1 Illustration of ST Distribution Shifts

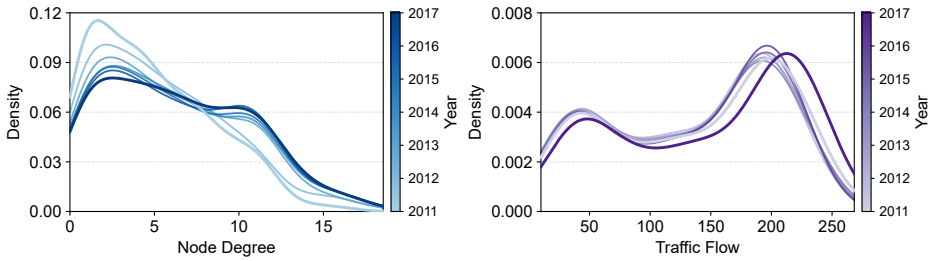

Figure 7: Evolution of node degree and week patterns across periods on PEMS-Stream.

We take PEMS-Stream dataset for instance which records traffic flow spanning seven years and the number of traffic sensors is gradually increased. As shown in Figure 7, we draw the Kernal Density Estimation (KDE) curve of traffic sensor node degree, which reflects the adjacency of nodes, in each year. At the initial year 2011, the low and high degrees account for the high and low densities, respectively. With the expansion of traffic sensor over years, the density of high node degree is increased. The last year 2017 reachs almost the extremums of both low and high degrees. This is reasonable since the traffic capacity is positively related to the scale of road networks. We also consider the evolving distribution of traffic flow. We firstly calculate the sum of traffic flow from all sensors as the road network traffic flow, and then regard one week as a basic unit computing the average flow of each time slot. From the KDE curves, we can know that the densities of traffic flow in each year are different, especially around the value of 200. Besides, the distribution curve of last year 2017 has an apparent right shift compared to other years, which indicates a relatively large increase of traffic flow in that year.

A.2 Details of Spatial and Temporal Modeling

A.2.1 Analysis of the Constant Basic Item

We analyze the role of the constant basic item α in the exponential kernel. Let

$$\mathbf{S} = \mathbf{E}_{\text{recv}}\mathbf{E}_{\text{send}}^{\top} \in \mathbb{R}^{N\times N}, \tag{17}$$

where S_{ij} denotes the similarity between receiving node i and sending node j. The kernel matrix and its row-normalized form are:

$$K_{ij} = \alpha + \exp(S_{ij}), \qquad A_{ij} = \frac{\alpha + \exp(S_{ij})}{\sum_{k=1}^{N}(\alpha + \exp(S_{ik}))}. \tag{18}$$

When $\alpha = 0$,

$$A_{ij} = \frac{\exp(S_{ij})}{\sum_{k=1}^{N}\exp(S_{ik})}, \tag{19}$$

Thus,

$$\mathbf{A} = \text{Softmax}(\mathbf{S}). \tag{20}$$

The kernel reduces exactly to the standard dot-product softmax adjacency.

When $\alpha = 1$,

$$\begin{aligned} A_{ij} &= \frac{1 + \exp(S_{ij})}{\sum_{k=1}^{N}(1 + \exp(S_{ik}))} \\ &= \frac{\exp\left(\text{Softplus}(S_{ij})\right)}{\sum_{k=1}^{N}\exp\left(\text{Softplus}(S_{ik})\right)}, \end{aligned} \tag{21}$$

Thus,

$$\mathbf{A} = \text{Softmax}\left(\text{Softplus}(\mathbf{S})\right). \tag{22}$$

Since

$$\mathrm{Softplus}(s) = \mathrm{ReLU}(s) + \log\left(1 + \exp(-|s|)\right), \tag{23}$$

and the second term approaches zero as $|s|$ increases, Softplus provides a smooth approximation to ReLU. Thus, Eq. equation 22 can be regarded as a smooth surrogate of

$$\mathbf{A} = \mathrm{Softmax}\left(\mathrm{ReLU}(\mathbf{S})\right). \tag{24}$$

More generally, for $\alpha > 0$,

$$\log\left(\alpha + \exp(s)\right) = \log\alpha + \mathrm{Softplus}\left(s - \log\alpha\right), \tag{25}$$

the normalized kernel can be written as:

$$\mathbf{A} = \mathrm{Softmax}\left(\mathrm{Softplus}\left(\mathbf{S} - \log\alpha\right)\right). \tag{26}$$

Therefore, $\alpha = 0$ recovers the standard dot-product softmax adjacency, while $\alpha = 1$ gives a Softplus-based approximation to the ReLU-based adaptive adjacency. Varying α provides additional flexibility in modeling shared basic correlations among node pairs.

A.2.2 Details of Kernel Features

Given two node embeddings $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{d_e}$, the exponential kernel with a constant basic item α is defined as:

$$\kappa(\mathbf{u}, \mathbf{v}) = \alpha + \exp(\mathbf{u}^\top \mathbf{v}), \qquad \alpha \geq 0. \tag{27}$$

In order to avoid explicitly computing kernel scores of all node pairs in the exponential kernel, we sample beforehand and fix the random projection vectors:

$$\mathbf{r}_a \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_e}), \quad a = 1, \ldots, m, \tag{28}$$

we then build positive random feature mapping as follows:

$$\begin{aligned} \phi_a(\mathbf{e}) &= \frac{1}{\sqrt{m}} \exp\left(\mathbf{r}_a^\top \mathbf{e} - \frac{\|\mathbf{e}\|_2^2}{2}\right), \\ \boldsymbol{\psi}(\mathbf{e}) &= \left[\sqrt{\alpha}, \phi_1(\mathbf{e}), \ldots, \phi_m(\mathbf{e})\right]^\top \in \mathbb{R}^{m+1}. \end{aligned} \tag{29}$$

The receiver and sender of nodes utilize the same random projections which are fixed during training. The constant feature $\sqrt{\alpha}$ corresponds to the basic item in the exponential kernel. According to Performer (Choromanski et al., 2020):

$$\begin{aligned} &\mathbb{E}_{\mathbf{r}}\left[\exp\left(\mathbf{r}^\top \mathbf{u} - \frac{\|\mathbf{u}\|^2}{2}\right) \exp\left(\mathbf{r}^\top \mathbf{v} - \frac{\|\mathbf{v}\|^2}{2}\right)\right] \\ &= \exp\left(-\frac{\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2}{2}\right) \mathbb{E}_{\mathbf{r}}\left[\exp\left(\mathbf{r}^\top (\mathbf{u} + \mathbf{v})\right)\right] \\ &= \exp\left(\frac{\|\mathbf{u} + \mathbf{v}\|^2 - \|\mathbf{u}\|^2 - \|\mathbf{v}\|^2}{2}\right) \\ &= \exp\left(\mathbf{u}^\top \mathbf{v}\right). \end{aligned} \tag{30}$$

Thus,

$$\phi(\mathbf{u})^\top \phi(\mathbf{v}) = \frac{1}{m} \sum_{a=1}^{m} \exp\left(\mathbf{r}_a^\top \mathbf{u} - \frac{\|\mathbf{u}\|^2}{2}\right) \exp\left(\mathbf{r}_a^\top \mathbf{v} - \frac{\|\mathbf{v}\|^2}{2}\right) \approx \exp(\mathbf{u}^\top \mathbf{v}). \tag{31}$$

Accordingly, we can use $\boldsymbol{\psi}$ to approximate exponential kernel $\kappa(\mathbf{u}, \mathbf{v})$ as follows:

$$\widehat{\kappa}(\mathbf{u}, \mathbf{v}) = \boldsymbol{\psi}(\mathbf{u})^\top \boldsymbol{\psi}(\mathbf{v}) = \alpha + \sum_{a=1}^{m} \phi_a(\mathbf{u}) \phi_a(\mathbf{v}) \approx \kappa(\mathbf{u}, \mathbf{v}). \tag{32}$$

A.2.3 Parameter Efficiency of Discrete Low-rank Embedding

We analyze the parameter efficiency of the discrete low-rank embedding compared with a full discrete embedding over the same weekly time slots. Let S denote the number of time slots per week, D the time embedding dimension, and r the low-rank dimension. A full discrete embedding maintains a learnable embedding dictionary $\mathbf{E} \in \mathbb{R}^{S \times D}$, requiring:

$$P_{\text{full}} = SD, \tag{33}$$

parameters. In contrast, the discrete low-rank embedding uses a time factor matrix $\mathbf{F} \in \mathbb{R}^{S \times r}$ and a shared linear projection from r to D dimensions. Including the projection bias, its parameter count is:

$$P_{\text{low}} = Sr + rD + D, \tag{34}$$

where the three terms correspond to the time factor matrix, projection weights, and projection bias, respectively. The relative parameter count and parameter reduction ratio are:

$$\rho = \frac{P_{\text{low}}}{P_{\text{full}}} = \frac{r}{D} + \frac{r+1}{S}, \tag{35}$$

$$\eta = 1 - \rho. \tag{36}$$

Thus, when $r \ll \min(S, D)$, the low-rank parameterization substantially reduces the number of trainable parameters by sharing the projection across all time slots. For instance, consider traffic data sampled every 5 minutes, yielding $S = 7 \times 288 = 2016$ weekly time slots. With $D = 128$ and $r = 8$, the parameter counts are:

$$P_{\text{full}} = 2016 \times 128 = 258{,}048, \tag{37}$$

$$P_{\text{low}} = 2016 \times 8 + 8 \times 128 + 128 = 17{,}280. \tag{38}$$

Thus, the resulting parameter reduction is:

$$\eta = 1 - \frac{17{,}280}{258{,}048} \approx 93.3\%. \tag{39}$$

A.3 Details of Experimental Setup

A.3.1 Datasets

In this paper, we use three real-world continuous traffic flow datasets: PEMS-Stream, PEMS06-Stream, and PEMS12-Stream. The statistics of these datasets are provided in Table 3. PEMS-Stream dataset is introduced in TrafficStream (Chen et al., 2021) and widely adopted in continual spatio-temporal forecasting. The initial name of this dataset is PEMSD3-Stream and the name of PEMS-Stream is used in recent literatures. PEMS06-Stream and PEMS12-Stream are introduced in this paper. However, the collection and processing of these two datasets are conducted in EvoXXL-Traffic (Yin et al., 2026), which are named as PEMS06 and PEMS12, respectively. For PEMS06 dataset, we select successive eleven years data (2015-2025) and extract three months data (1st, June to 31st, August) for each year, which derives our used PEMS06-Stream dataset in this paper. The procedure of PEMS12 dataset is similar which spans successive four years data (2015-2018), forming PEMS12-Stream dataset.

Table 3: Statistics of three continuous traffic flow datasets.

Dataset	Time Span	Period	Node Expansion	Frequency
PEMS-Stream	07/10/2011–09/08/2017	7	655→715→786→822→ 834→850→871	5 min
PEMS06-Stream	06/01/2015–08/31/2025	11	418→461→505→558→ 624→630→669→670→ 690→705→746	5 min
PEMS12-Stream	06/01/2015–08/31/2018	4	2365→2413→ 2442→2538	5 min

A.3.2 BASELINES

We compare our method to eight baselines. The brief introductions are as follows. ❶ Retrain-ST: Basic STGNN (Chen et al., 2021) with retrain strategy. In each period, the model is trained from scratch and the model weights are not used in subsequent periods. ❷ Pretrain-ST: Basic STGNN with pretrain strategy. The model is trained in the first period and the model weights are directly used in subsequent all periods. Since the basic STGNN is node-agnostic, thus this strategy can be achieved. ❸ Online-ST-AN: Basic STGNN with strategy of online learning on all nodes of each period. In each period, the model is initialized using model weights from last period and then finetuned using data from all nodes, including the existing nodes and newly added nodes. In the first period, the model is trained from scratch. ❹ Online-ST-NN: Basic STGNN with strategy of online learning on newly added nodes of each period. The new nodes combined with their neighbor nodes are used to build subgraphs. In each period, the model is initialized using model weights from last period and then finetuned using data from newly added nodes. In the first period, the model is trained from scratch. The advantage of this strategy is that training merely on data from newly added nodes is more efficient than training on all nodes since the counts of new nodes are much less than all nodes. However, it also incurs catastrophic forgetting that the knowledge learned on new nodes are not applicable to existing nodes. ❺ TrafficStream (Chen et al., 2021): adopts the strategy of training on newly added nodes for learning new knowledge with high efficiency. Besides, to further absorb the new patterns, it detects and incorporates the influential existing nodes whose traffic patterns change greatly. Meanwhile, to retain the learned knowledge of existing nodes and alleviate catastrophic forgetting, it detects and incorporates the representative existing nodes as a replay whose traffic patterns are stable. Thus, the old knowledge of existing nodes are maintained. Furthermore, it utilizes elastic weight consolidation (EWC) to protect the important trained parameters. ❻ STKEC (Wang et al., 2023a): It also trains models on newly added nodes and uses influence function to detect and incorporate important nodes for learning new knowledge. Meanwhile, it uses explicit memory module and Fisher regularization to retain the old knowledge. ❼ EAC (Chen & Liang, 2025): It uses learnable prompt to adapt to the evolving distribution while freezing the backbone to retain the learned knowledge. ❽ STBP (Liu & Zhang, 2026): It uses contextual pattern bank for the gating and attention mechanisms to learn new knowledge while freezing the backbone to retain the learned knowledge.

A.3.3 EVALUATION METRICS

We adopt three commonly used metrics to evaluate our method and baselines: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). MAE computes the absolute differences between predictions and ground truths, reflecting the general deviations. RMSE and MAPE are sensitive to large and small values, respectively. The formulations of these metrics are as follows where $\hat{y}_i$ and y_i refer to the prediction and ground truth, respectively, n is the number of samples.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{40}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{41}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \times 100\% \tag{42}$$

A.3.4 IMPLEMENTATION

EvoST contains an input embedding layer, a time embedding layer, a linearized kernel graph convolution layer, an MLP layer, and an output layer. The input embedding layer transforms traffic state data into high dimensional representaions using 1D convolution. The kernel size equals to the sequence length of data, thus a temporal patch (Nie et al., 2023) is formed whose time step equals to 1. The input dimension is set to 64 for all datasets. A time embedding layer encodes current time slots into discrete low-rank embedding and continuous phase embedding which are then added to form the weekly time embedding. The input data embedding and weekly time embedding are

Table 4: Performance comparison across different periods on PEMS-Stream. Best results are highlighted in **red bold**, and second-best results are marked with blue underlines.

Dataset		*PEMS-Stream* (655 → ··· → 871)							
Method	**Metric**	**2011**	**2012**	**2013**	**2014**	**2015**	**2016**	**2017**	**Avg.**
Retrain-ST	MAE	16.19±0.05	15.40±0.04	15.13±0.04	15.54±0.06	15.38±0.02	15.13±0.02	16.99±0.04	15.68±0.01
	RMSE	25.72±0.10	25.02±0.06	24.45±0.11	25.76±0.13	25.98±0.05	27.61±0.07	28.24±0.09	26.11±0.02
	MAPE(%)	19.91±0.69	20.06±0.92	19.68±0.77	21.18±0.72	20.25±0.62	19.63±0.35	20.46±0.88	20.17±0.22
Pretrain-ST	MAE	16.19±0.05	15.47±0.04	15.29±0.04	15.75±0.02	15.65±0.03	15.51±0.05	17.32±0.06	15.89±0.04
	RMSE	25.72±0.10	25.12±0.06	24.73±0.09	26.30±0.08	26.61±0.07	28.38±0.15	29.02±0.13	26.55±0.07
	MAPE(%)	19.91±0.69	19.78±0.69	20.35±0.78	22.30±0.66	21.76±0.68	20.71±0.65	22.37±1.28	21.03±0.71
Online-ST-AN	MAE	16.19±0.05	15.26±0.05	15.01±0.03	15.40±0.03	15.25±0.01	14.98±0.02	16.79±0.03	15.56±0.02
	RMSE	25.72±0.10	24.75±0.07	24.22±0.08	25.45±0.09	25.75±0.04	27.26±0.05	27.81±0.05	25.85±0.04
	MAPE(%)	19.91±0.69	19.60±0.50	20.04±0.80	20.51±0.51	20.53±0.71	19.57±1.07	20.15±0.50	20.05±0.30
Online-ST-NN	MAE	16.19±0.05	17.29±0.30	16.11±0.04	17.72±0.36	72.32±46.09	20.48±0.88	41.02±14.15	28.73±7.37
	RMSE	25.72±0.10	27.93±0.44	26.73±0.14	30.73±0.99	186.61±129.65	39.14±2.15	103.00±58.97	62.84±20.70
	MAPE(%)	19.91±0.69	24.42±2.97	21.25±0.42	22.41±0.71	64.73±30.72	27.33±2.28	29.73±4.65	29.97±5.11
TrafficStream	MAE	16.19±0.05	16.07±0.14	15.65±0.06	15.99±0.07	15.95±0.21	15.65±0.12	17.57±0.18	16.15±0.08
	RMSE	25.72±0.10	26.32±0.57	25.65±0.23	26.80±0.23	27.57±0.70	28.73±0.43	29.42±0.40	27.17±0.30
	MAPE(%)	19.91±0.69	21.20±1.77	20.17±0.48	21.69±1.13	20.77±0.40	20.04±0.51	20.30±0.44	20.58±0.39
STKEC	MAE	16.21±0.03	15.96±0.11	15.82±0.26	16.06±0.13	15.94±0.12	15.74±0.09	17.58±0.04	16.19±0.07
	RMSE	25.77±0.07	25.94±0.23	26.15±0.78	28.20±2.49	27.45±0.87	28.87±0.43	29.40±0.10	27.40±0.51
	MAPE(%)	19.50±0.33	20.34±0.75	20.56±0.93	21.33±0.79	21.84±1.09	19.94±0.29	21.14±0.80	20.66±0.22
EAC	MAE	15.71±0.14	14.17±0.02	13.93±0.04	14.31±0.04	14.33±0.04	13.96±0.05	15.82±0.05	14.61±0.04
	RMSE	24.85±0.40	22.59±0.07	22.27±0.07	23.43±0.05	24.05±0.07	25.70±0.15	26.17±0.09	24.15±0.09
	MAPE(%)	19.19±0.39	18.72±0.26	19.13±0.19	20.21±0.42	20.00±0.52	19.04±0.51	20.49±0.37	19.54±0.15
STBP	MAE	14.28±0.63	11.84±0.14	11.66±0.09	11.89±0.14	11.82±0.14	11.38±0.16	13.21±0.12	12.30±0.15
	RMSE	22.63±1.41	19.07±0.27	18.78±0.16	19.79±0.21	20.09±0.23	22.01±0.17	22.22±0.16	20.66±0.28
	MAPE(%)	16.81±0.60	**15.15**±0.10	15.43±0.09	**16.15**±0.17	**15.74**±0.15	14.79±0.19	**16.16**±0.12	15.75±0.13
Ada-ST	MAE	14.51±0.29	12.69±0.33	12.35±0.06	12.53±0.09	12.73±0.07	12.21±0.06	14.12±0.12	13.02±0.07
	RMSE	22.68±0.48	20.10±0.28	19.50±0.09	20.63±0.17	21.48±0.13	23.13±0.15	23.31±0.20	21.55±0.07
	MAPE(%)	18.15±0.65	18.31±2.14	18.32±1.72	17.38±0.41	18.07±0.60	17.42±0.97	18.00±0.57	17.95±0.61
EvoST	MAE	**13.09**±0.12	**11.57**±0.04	**11.38**±0.07	**11.58**±0.04	**11.50**±0.05	**10.87**±0.05	**12.82**±0.07	**11.83**±0.05
	RMSE	**20.62**±0.40	**18.71**±0.05	**18.43**±0.12	**19.28**±0.08	**19.55**±0.11	**21.22**±0.08	**21.47**±0.10	**19.89**±0.08
	MAPE(%)	**16.14**±0.25	15.26±0.23	**15.40**±0.22	**16.15**±0.14	16.00±0.18	**14.63**±0.14	16.26±0.41	**15.69**±0.14

concatenated along the feature dimension. A linearized kernel graph convolution layer further encodes the embeddings for spatial correlations modeling, followed by an MLP layer to encode in the feature dimensions. Residual connection, layer normalization and dropout are also applied. Finally, the encoded embeddings are projected through an output layer to obtain the predicted future traffic states. EvoST utilizes AdamW optimizer to update model parameters. The learning rate is set to 0.002 and will be halved at the 1st, 20th, 30th, 40th epoch. The maximum epochs are set to 100 and the early stopping strategy is employed with patience of 30 epochs. We use PyTorch 1.12.1 to implement EvoST. All experiments are conducted on NVIDIA A100 40GB GPU using uniform code framework and run five times with random seeds ranging from 2023 to 2027. The average and standard deviation of five runs are reported.

A.4 Additional Experimental Results

A.4.1 Performance Comparisons in Each Period

We report the prediction performances of all methods in each period of PEMS-Stream, PEMS06-Stream, and PEMS12-Stream datasets, which are provided in Table 4, 5, and 6, respectively.

A.4.2 Ablation Study

We conduct ablation study on PEMS06-Stream dataset whose results are similar to those of PEMS-Stream. As shown in Figure 8, the worse performance of variant w/ retrain indicates the necessity of knowledge transferring across periods. The variants w/o gcn and w/o time perform worse than

Table 5: Performance comparison across different periods on PEMS06-Stream. Best results are highlighted in **red bold**, and second-best results are marked with blue underlines.

Dataset		*PEMS06-Stream* (418 → ⋯ → 746)											
Method	**Metric**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**2021**	**2022**	**2023**	**2024**	**2025**	**Avg.**
Retrain-ST	MAE	13.02±0.04	14.21±0.08	15.91±0.18	14.35±0.05	13.86±0.06	11.06±0.08	13.26±0.09	12.66±0.05	13.32±0.09	13.62±0.05	13.85±0.07	13.56±0.02
	RMSE	21.06±0.09	23.65±0.20	27.04±0.38	23.25±0.11	22.20±0.16	17.46±0.10	22.28±0.18	21.42±0.15	22.03±0.23	22.89±0.18	23.00±0.19	22.39±0.04
	MAPE(%)	21.65±1.07	21.13±0.49	19.45±0.40	21.77±1.84	18.95±0.14	18.50±0.79	19.05±0.73	19.71±0.78	18.81±0.80	17.71±0.38	18.02±0.29	19.52±0.14
Pretrain-ST	MAE	13.02±0.04	14.38±0.03	16.14±0.05	14.78±0.08	14.33±0.05	11.50±0.06	13.82±0.13	13.25±0.07	14.09±0.04	14.32±0.06	14.54±0.07	14.01±0.02
	RMSE	21.06±0.09	23.77±0.11	27.41±0.10	23.69±0.12	22.72±0.07	18.12±0.09	22.87±0.16	22.69±0.25	23.56±0.22	24.09±0.21	24.45±0.21	23.13±0.11
	MAPE(%)	21.65±1.07	23.15±1.29	21.69±0.97	28.50±4.46	26.25±4.00	21.00±1.54	29.62±7.20	22.57±2.35	24.06±2.66	24.78±4.14	21.25±1.38	24.05±2.60
Online-ST-AN	MAE	13.02±0.04	14.09±0.06	15.82±0.17	14.19±0.08	13.74±0.05	10.88±0.06	13.11±0.05	12.40±0.04	13.20±0.05	13.45±0.04	13.72±0.03	13.42±0.02
	RMSE	21.06±0.09	23.32±0.10	26.88±0.40	23.01±0.17	21.89±0.10	17.17±0.08	21.96±0.12	21.07±0.07	21.80±0.10	22.49±0.09	22.75±0.17	22.13±0.07
	MAPE(%)	21.65±1.07	21.57±1.44	19.65±0.53	20.79±0.67	19.84±0.82	17.98±0.86	19.63±0.79	18.10±0.65	19.67±1.22	17.91±0.85	18.04±0.15	19.53±0.16
Online-ST-NN	MAE	13.02±0.04	14.88±0.09	16.62±0.16	14.89±0.05	14.46±0.14	11.80±0.14	13.91±0.11	13.96±0.15	16.08±0.23	15.69±0.49	14.77±0.27	14.55±0.08
	RMSE	21.06±0.09	25.16±0.31	28.96±0.86	24.14±0.13	22.97±0.16	18.27±0.08	23.78±0.82	24.04±0.89	29.06±0.72	29.56±7.40	24.38±0.33	24.67±0.93
	MAPE(%)	21.65±1.07	23.14±1.13	21.45±0.87	22.90±0.94	22.71±1.92	23.41±1.98	23.20±2.10	25.16±2.61	31.83±6.33	29.22±5.37	27.58±6.12	24.75±1.73
TrafficStream	MAE	13.02±0.04	14.45±0.10	16.14±0.10	14.59±0.08	14.08±0.13	11.28±0.17	13.58±0.11	12.75±0.09	13.51±0.16	13.84±0.07	14.25±0.25	13.77±0.04
	RMSE	21.06±0.09	24.01±0.28	27.30±0.22	23.64±0.24	22.43±0.42	18.43±1.27	23.93±1.57	21.41±0.21	22.49±0.61	23.43±0.64	24.23±2.09	22.94±0.37
	MAPE(%)	21.65±1.07	21.68±0.84	20.88±0.83	22.66±0.62	20.57±0.85	18.92±0.67	20.99±1.54	19.79±1.31	20.74±1.92	18.71±0.64	19.78±1.35	20.58±0.43
STKEC	MAE	12.91±0.07	14.40±0.09	16.20±0.16	14.62±0.09	14.14±0.05	11.17±0.03	13.49±0.11	12.76±0.09	13.60±0.08	13.89±0.20	14.13±0.08	13.76±0.05
	RMSE	20.92±0.13	24.08±0.22	27.40±0.39	23.54±0.16	22.65±0.19	17.63±0.12	22.55±0.24	21.50±0.16	23.39±0.73	24.50±1.71	23.33±0.24	22.86±0.18
	MAPE(%)	**20.99**±0.63	21.60±0.74	21.03±1.35	23.06±1.51	20.99±1.23	18.81±0.35	20.90±0.96	19.67±1.31	20.48±1.09	19.64±1.35	19.96±1.30	20.65±0.47
EAC	MAE	12.54±0.16	13.71±0.10	19.85±0.99	13.95±0.09	13.65±0.08	10.88±0.08	12.95±0.08	12.34±0.04	13.05±0.15	13.62±0.12	13.48±0.08	13.64±0.13
	RMSE	19.96±0.28	22.68±0.23	35.00±3.04	22.45±0.16	21.60±0.19	17.18±0.12	21.54±0.21	20.83±0.11	22.17±1.57	22.98±0.25	22.34±0.10	22.61±0.48
	MAPE(%)	23.16±1.48	22.66±0.63	24.54±1.76	26.61±2.61	24.51±2.73	19.68±1.18	26.43±4.05	21.86±1.61	23.18±2.40	22.78±2.85	20.31±1.25	23.25±1.65
STBP	MAE	13.74±1.15	12.45±0.16	15.66±0.49	12.45±0.18	12.38±0.31	10.66±1.02	11.83±0.12	11.24±0.16	13.03±2.72	12.43±0.33	12.71±0.19	12.60±0.33
	RMSE	50.65±18.51	20.80±0.27	32.66±4.88	20.34±0.49	22.02±3.63	48.77±44.02	23.01±2.83	31.05±16.99	95.14±152.44	27.96±16.01	27.49±8.76	36.35±19.05
	MAPE(%)	82.05±34.39	**18.86**±0.33	21.33±2.43	18.64±0.42	18.27±1.03	25.52±11.86	**16.97**±0.25	18.24±1.18	36.35±39.20	**16.00**±0.46	18.18±1.46	26.40±3.57
Ada-ST	MAE	12.38±0.15	13.20±0.14	17.26±1.34	13.26±0.15	12.98±0.16	10.62±0.20	12.43±0.13	11.66±0.08	12.46±0.22	13.16±0.14	12.90±0.13	12.94±0.15
	RMSE	19.92±0.23	21.71±0.28	30.25±2.88	21.43±0.42	20.43±0.26	16.76±0.43	20.88±0.31	19.79±0.10	20.51±0.24	22.05±0.30	21.40±0.24	21.37±0.31
	MAPE(%)	25.13±2.40	20.67±0.99	21.72±3.34	20.24±1.00	19.24±1.57	18.46±1.82	17.95±0.24	17.10±0.87	18.94±1.75	18.74±2.11	18.00±0.42	19.66±0.46
EvoST	MAE	**10.78**±0.06	**12.15**±0.07	**15.29**±0.47	**12.17**±0.04	**11.98**±0.03	**9.40**±0.11	**11.56**±0.03	**10.90**±0.06	**11.59**±0.08	**11.91**±0.04	**12.20**±0.07	**11.81**±0.06
	RMSE	**17.95**±0.16	**20.03**±0.11	**26.45**±0.48	**19.65**±0.05	**18.89**±0.06	**15.10**±0.39	**19.36**±0.10	**18.31**±0.12	**18.77**±0.14	**19.45**±0.07	**19.86**±0.11	**19.44**±0.08
	MAPE(%)	21.71±3.08	19.42±0.17	**19.36**±1.49	**18.53**±0.13	**17.47**±0.08	**15.61**±0.13	17.25±0.25	**16.23**±0.22	**16.97**±0.17	16.07±0.07	**16.46**±0.19	**17.74**±0.37

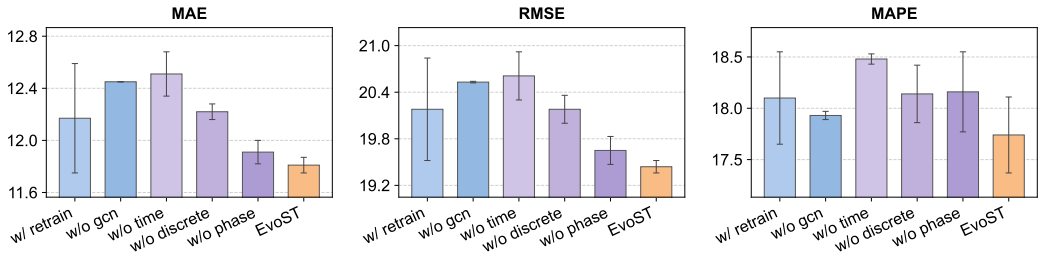

Figure 8: Ablation study on PEMS06-Stream.

EvoST, which reflects the essential functions of the spatial and temporal modules in EvoST. The variants w/o discrete and w/o phase further show the contributions of the discrete low-rank embedding and continuous phase embedding.

A.4.3 Hyperparameter Sensitivity Analysis

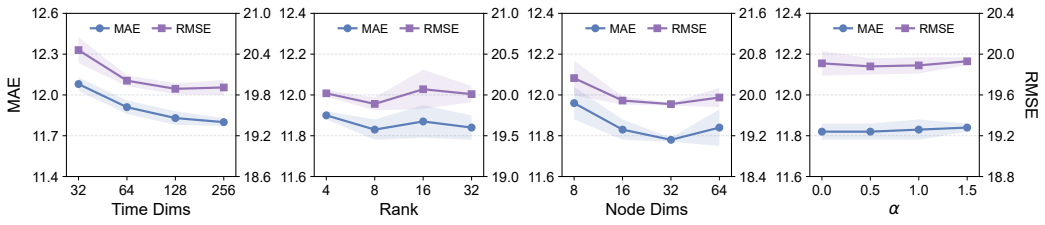

Figure 9: Analysis of four hyperparameters on PEMS-Stream.

We investigate the sensitivity of hyperparameters on PEMS-Stream dataset, as shown in Figure 9. For the time dimension, the prediction errors drop fast from 32 to 128. At the dimension of 256, the

Table 6: Performance comparison across different periods on PEMS12-Stream. Best results are highlighted in **red bold**, and second-best results are marked with blue underlines.

Dataset		*PEMS12-Stream* (2365 → · · · → 2538)				
Method	**Metric**	**2015**	**2016**	**2017**	**2018**	**Avg.**
Retrain-ST	MAE	15.46±0.08	15.63±0.09	15.76±0.11	15.21±0.07	15.51±0.06
	RMSE	28.78±0.18	28.00±0.21	28.31±0.23	27.63±0.09	28.18±0.12
	MAPE(%)	25.52±0.43	25.75±1.50	25.97±1.57	25.34±1.40	25.65±0.59
Pretrain-ST	MAE	15.46±0.08	15.76±0.09	16.18±0.12	15.23±0.10	15.66±0.08
	RMSE	28.78±0.18	28.09±0.16	28.52±0.18	27.63±0.13	28.25±0.14
	MAPE(%)	25.52±0.43	33.45±4.24	45.19±10.13	26.35±1.11	32.63±3.85
Online-ST-AN	MAE	15.46±0.08	15.54±0.06	15.61±0.07	15.10±0.07	15.42±0.05
	RMSE	28.78±0.18	27.81±0.13	27.98±0.13	27.42±0.04	28.00±0.07
	MAPE(%)	25.52±0.43	26.71±1.72	26.42±1.86	25.80±1.48	26.11±0.78
Online-ST-NN	MAE	15.46±0.08	16.53±0.06	17.18±0.23	16.81±0.41	16.49±0.18
	RMSE	28.78±0.18	29.96±0.17	30.94±0.49	31.07±1.02	30.19±0.40
	MAPE(%)	25.52±0.43	30.67±2.31	29.25±1.82	26.36±0.48	27.95±0.68
TrafficStream	MAE	15.46±0.08	15.64±0.03	15.75±0.07	15.30±0.07	15.54±0.04
	RMSE	28.78±0.18	28.08±0.09	28.22±0.15	27.86±0.19	28.23±0.09
	MAPE(%)	25.52±0.43	27.53±2.03	26.63±0.84	25.65±1.16	26.33±0.72
STKEC	MAE	15.37±0.04	15.55±0.03	15.81±0.06	15.29±0.03	15.50±0.03
	RMSE	28.56±0.10	27.92±0.04	28.23±0.09	27.82±0.18	28.13±0.06
	MAPE(%)	26.04±2.59	27.46±1.60	28.97±2.74	**25.00**±0.88	26.87±0.63
EAC	MAE	14.29±0.10	14.32±0.05	14.71±0.12	14.47±0.15	14.44±0.07
	RMSE	26.32±0.16	25.43±0.15	26.32±0.73	25.80±0.31	25.97±0.22
	MAPE(%)	25.25±1.73	30.68±2.41	42.32±7.00	27.42±1.12	31.42±2.23
STBP	MAE	13.70±0.46	13.61±0.23	13.78±0.15	14.00±0.58	13.77±0.18
	RMSE	39.55±21.62	25.31±1.25	28.63±1.63	32.12±14.67	31.40±5.81
	MAPE(%)	24.43±1.79	24.83±1.97	26.52±4.05	26.63±2.21	25.60±0.80
Ada-ST	MAE	14.00±0.24	14.27±0.17	14.01±0.22	14.29±0.10	14.14±0.12
	RMSE	25.89±0.45	25.13±0.38	24.88±0.20	26.08±0.79	25.50±0.19
	MAPE(%)	25.46±2.08	26.30±4.42	25.92±1.28	26.44±2.57	26.03±1.94
EvoST	MAE	**12.94**±0.12	**12.80**±0.02	**12.76**±0.05	**13.05**±0.23	**12.89**±0.09
	RMSE	**24.72**±1.01	**22.93**±0.08	**22.92**±0.26	**25.75**±4.75	**24.08**±1.48
	MAPE(%)	**22.84**±0.29	**22.78**±0.17	**22.63**±0.58	26.48±0.56	**23.69**±0.22

MAE gains little extra benefits and the RMSE gets worser. The optimal rank is at 8 after which the model will have larger prediction errors and instabilities. For the node dimension, the errors drop fast from 8 to 16 and get little gains until 32. The constant basic item α in kernel features is not sensitive on PEMS-Stream in the range of 0.0 to 1.5.

A.5 Additional Visualizations

A.5.1 Evolving Graph Structures and Temporal Patterns

We visualize the differences of learned graph adjacencies and temporal patterns across successive periods from 2014 to 2017, as shown in Figure 10. It can be seen that EvoST learns stable spatial adjacencies and temporal patterns while also adapts to the changing spatial and temporal distributions across periods.

A.5.2 Adaptive Adjacencies Involving New Nodes

We investigate how the newly added nodes affect the spatial relations as the graph network expansion. As shown in Figure 11 and Figure 12, the first column reflects the influences of old nodes to new nodes while the second column is about new nodes to old nodes. Ths last column shows the influences between new nodes themselves. It can be seen that the newly added nodes are not isolated and have connections with old nodes. Several old nodes which may be traffic hubs have strong influences on some of new nodes. Similarly, the new nodes can also affect old nodes significantly. The connections between new nodes themselves can also be formed. Besides, the new adjacencies also

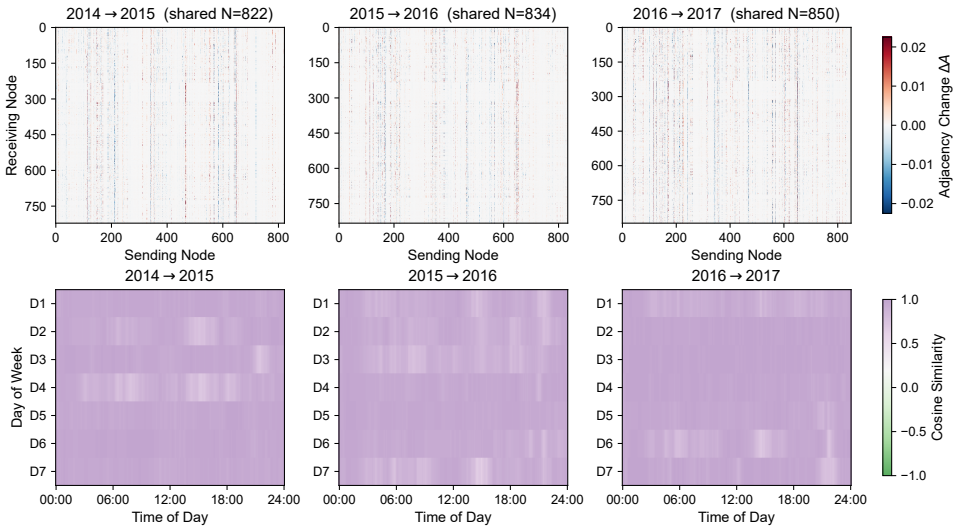

Figure 10: Evolving graph structures and temporal patterns across periods 2014 to 2017.

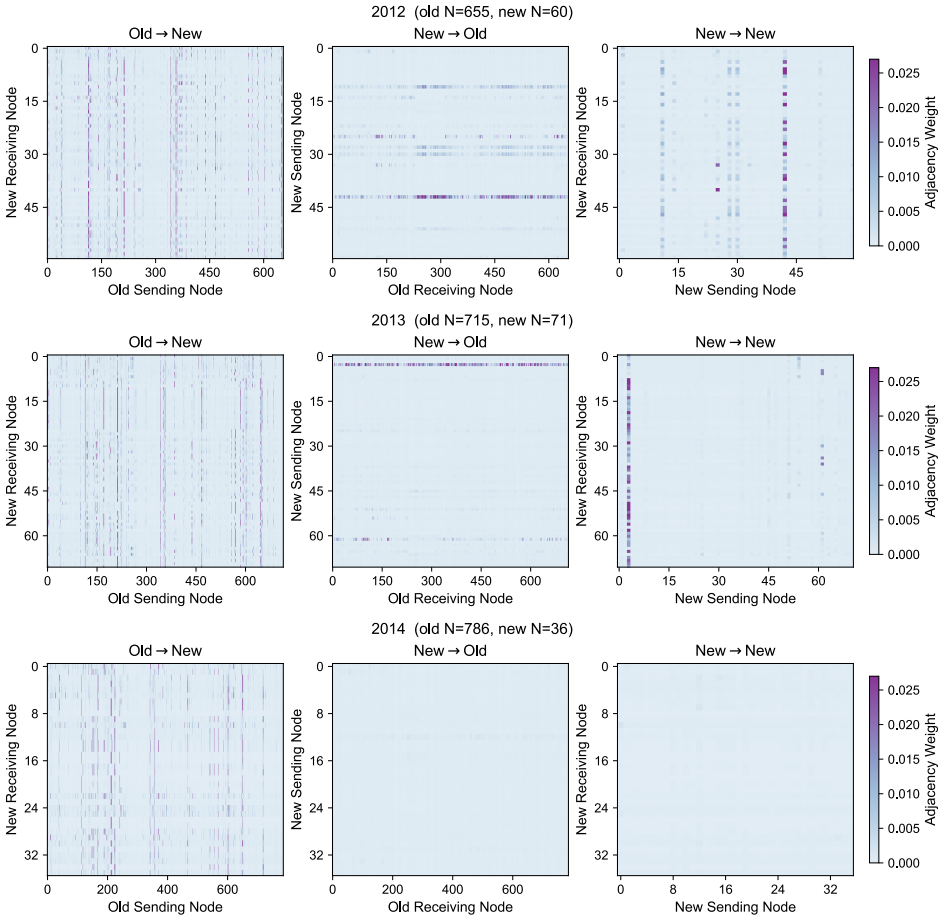

Figure 11: Adjacencies involving newly added nodes in the period of 2012, 2013, and 2014

exhibit heterogeneities in different periods. For instance, several new nodes have strong impacts on old nodes in the period 2012 while the new nodes in period 2014 have much weak impacts. These

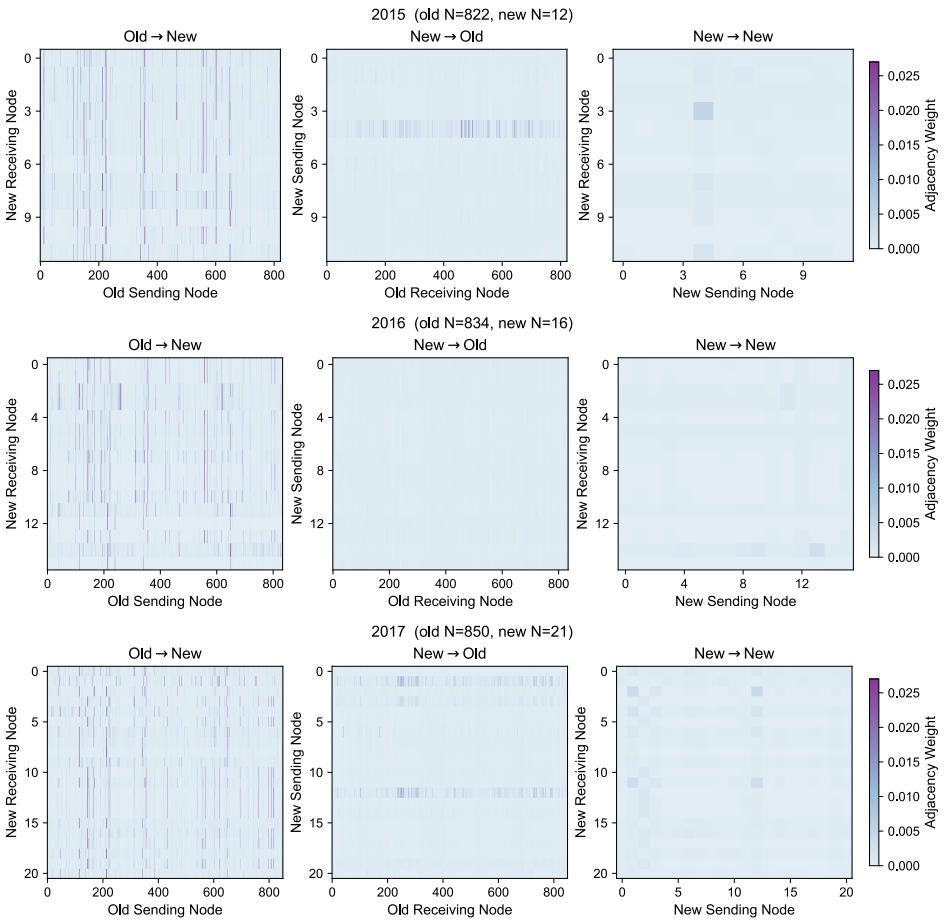

Figure 12: Adjacencies involving newly added nodes in the period of 2015, 2016, and 2017

illustrations indicate that the traffic network expansion changes the spatial distributions and EvoST is capable of adapting to them.

A.5.3 Embedding Trajectories in PCA Space

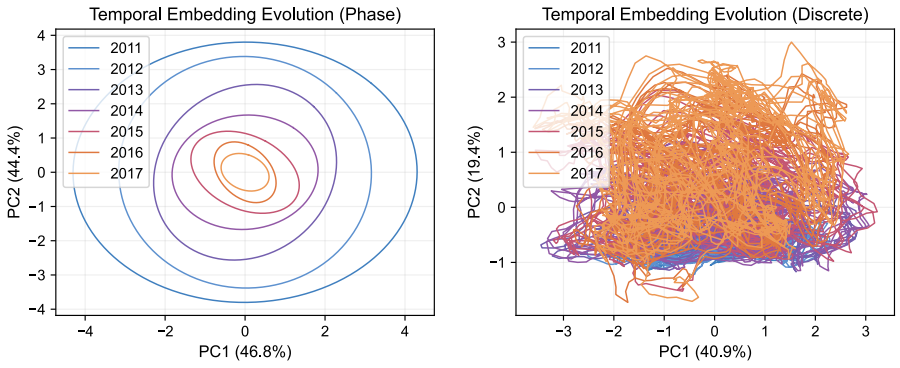

Figure 13: PCA of phase and discrete embeddings across different periods on PEMS-Stream.

We draw the trajectory of continuous phase embedding and discrete low-rank embedding separately in the shared PCA space, as shown in Figure 13. We can know that the phase embeddings have

significant periodic patterns and the evolving trends across successive periods. On the contrary, the discrete embeddings exhibit irregularities in the two-dimensional PCA space. Considering the trajectory patterns of time embeddings in Figure 6, we can know that the patterns of time embeddings are influenced under the collective effects of the two component embeddings.

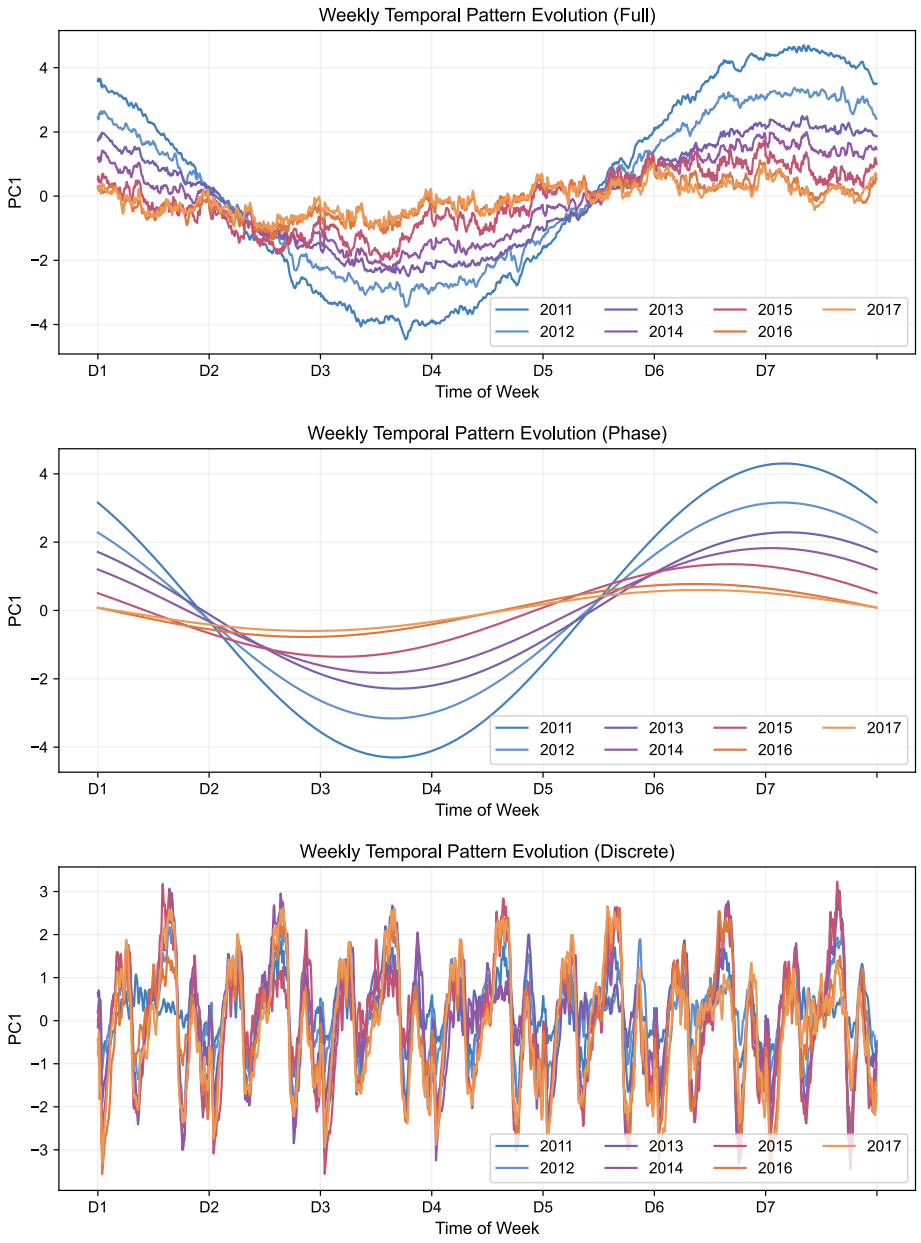

Figure 14: Weekly temporal pattern evolutions in PC1 space.

We then project the embeddings into the primary direction PC1. As shown in Figure 14, x-axis refers to the successive time slots of one week and y-axis refers to the PC1 value. It can be seen that the phase embedding exhibits pronounced weekly patterns which is corresponding to its patterns in two-dimensional PCA space. While for the discrete embedding, its patterns show regular fluctuations rather than the incomprehensibility in two-dimensional space. The patterns of the full time embeddings show the weekly periodicity while also have local fluctuations. It is apparent that the phase embedding and discrete embedding jointly influence the patterns of full time embeddings. Furthermore, the phase and discrete embeddings in PC1 space are highly similar to the components

of trend and event that are decomposed from traffic state series. From this perspective, the continuous phase embedding and discrete low-rank embedding have complementary functions which contribute to the effectiveness of the weekly time embeddings.