Expert Systems With Applications 244 (2024) 122962

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A generalized feature projection scheme for multi-step traffic forecasting

Adnan Zeb [a], Shiyao Zhang [a,b], Xuetao Wei [a], James Jianqiao Yu [a,c,*]

[a] Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
[b] Research Institute for Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen 518055, China
[c] Department of Computer Science, University of York, YO10 5DD York, UK

ARTICLE INFO

Keywords:
Intelligent transportation systems
Traffic forecasting
Feature projection
Spatial transformations

ABSTRACT

Exploiting spatial–temporal correlations has long been regarded as the cornerstone of traffic state prediction. Among existing techniques, temporal graph neural networks (TGNNs) have recently emerged as a prominent solution for modeling complex spatial–temporal traffic data correlations. Existing studies on TGNNs mainly focus on developing new building blocks to embed hidden correlations into a unified latent representation, which is mapped to predictions of distinct horizons. However, mapping the same latent features to distinct scalar predictions makes the gradient computation challenging for updating model parameters in the relevant directions. Besides, TGNNs are biased towards the shared temporal patterns while neglecting the complex dependencies within each data series, which can be captured to enrich latent features. To handle these problems jointly, we propose a novel feature projection scheme for the traffic prediction framework of TGNNs. The proposed projection scheme is based on spatial convolutions that first generate horizon-specific feature maps and then transform them into scalar predictions of the corresponding horizons. These horizon-specific feature maps establish interactions between the unified latent representation and the corresponding output values to bring the predictions closer to the true values. Besides, the proposed scheme also serves as a pattern modeling phase that enhances the expressivity of TGNNs by enriching latent features with data source-wise patterns of distinct time steps. Comprehensive experiments on two real-world traffic datasets demonstrate that the proposed scheme enhances the predictive performance and reduces the model parameters of TGNNs.

1. Introduction

With the rapid urbanization in the past few decades, intelligent transportation systems (ITS) have become an integral component of modern smart cities (Jiang & Luo, 2022; Lin et al., 2017). Recent advancements in data acquisition tools have raised the interest of the research community in developing data-driven approaches for solving traffic problems in the cities (Yu, 2022; Yu, Markos, & Zhang, 2021; Zhang et al., 2011). The data collection tools are usually deployed on road segments in different parts of the cities to collect time series traffic data, which are then analyzed for improving traffic management and congestion prevention strategies (Jia, Jiang, Liu, Cui, & Shi, 2017; Kong et al., 2013; Snyder & Do, 2019; Song, Lin, Guo, & Wan, 2020; Zhu, Yu, Wang, Ning, & Tang, 2018). Traffic state prediction has, therefore, become a significant research direction in time series forecasting, aiming at utilizing historical traffic state observations to predict future conditions and use them to avoid traffic problems in the cities.

Traffic data naturally inherits complex non-linear spatial–temporal relationships among data sources, which are required to be modeled for an accurate prediction of the future states (Baggag et al., 2021; Guo, Lin, Wan, Li, & Cong, 2022; Lablack & Shen, 2023; Yu, 2023). Several deep learning techniques have been recently developed for modeling traffic dependencies, among which temporal graph neural networks (TGNNs), particularly recurrent neural networks (RNNs) based TGNNs, have shown significant capabilities in modeling complex spatial–temporal dependencies. These TGNNs are mainly composed of three main modules, namely, a graph generation layer, a spatial–temporal convolutional layer, and a feature projection layer.

The graph generation layer constructs the adjacency matrix either based on a pre-defined graph structure (Guo, Lin, Feng, Song, & Wan, 2019; Huang, Huang, Liu, Dai, & Kong, 2021; Li, Yu, Shahabi, & Liu, 2018; Wang et al., 2020; Yu, Yin, & Zhu, 2018; Zheng, Fan, Wang, & Qi, 2020) or via adaptive strategies (Bai, Yao, Li, Wang, & Wang, 2021; Jin et al., 2022; Li et al., 2022; Wu et al., 2020; Wu, Pan, Long, Jiang, & Zhang, 2019). The spatial–temporal convolutional layer comprises a spatial and a temporal component, collaborating dynamically with each other. The former component utilizes the connectivity structure of data sources and the geographical distance between them to capture the spatial dependencies. The later component captures correlations among

* Corresponding author at: Department of Computer Science, University of York, YO10 5DD York, UK.
E-mail addresses: adnanzeb@sustech.edu.cn (A. Zeb), zhangsy@sustech.edu.cn (S. Zhang), weixt@sustech.edu.cn (X. Wei), jqyu@ieee.org (J.J. Yu).

https://doi.org/10.1016/j.eswa.2023.122962
Received 30 October 2023; Received in revised form 22 November 2023; Accepted 13 December 2023
Available online 16 December 2023

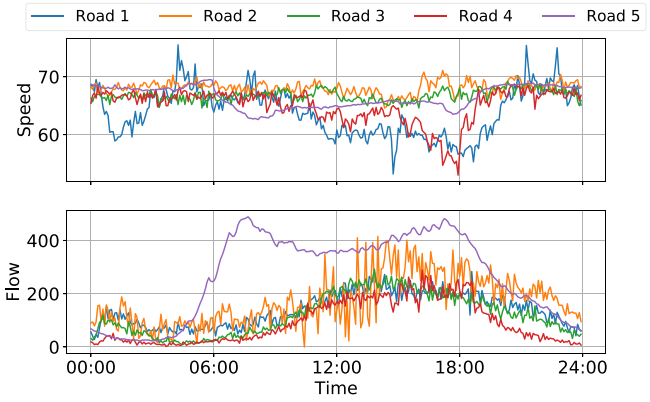

Fig. 1. Comparison of traffic speed and flow data for five random roads on the PeMSD4 dataset. Each road reveals a larger shift in flow values than speed in the given time period. Besides, each road exhibits distinct patterns at different time steps.

traffic readings recorded at different time steps. The feature projection layer linearly transforms a unified latent representation of historical state observations into the predictions of desired horizons either as a whole or auto-regressively through each horizon. Here, we consider the case where multi-step predictions are generated as a whole (Bai et al., 2021; Diao et al., 2019; Guo, Lin, Li, Chen, & Wan, 2019; Li et al., 2022; Liao et al., 2018; Lv et al., 2018; Wang, Chen, & He, 2019; Wu et al., 2020, 2019). This setting has recently been widely adopted in TGNNs since it resolves the problem of inconsistency between training and testing in traditional auto-regressive settings.

While TGNNs achieved promising results, multi-step traffic state prediction remains challenging due to the following key limitations. First, the regular feature projection layer in TGNNs mainly receives a unified latent representation of historical states from the spatial–temporal convolutional layer. The feature-to-prediction mapping mechanism in the regular feature projection layer maps the temporal dimension of input to the desired output dimension via either a fully connected layer or planner convolution (Bai et al., 2021; Wu et al., 2020, 2019). In the latter case, the desired output dimension Q is typically obtained by defining Q filters, which transform the unified latent representation into the predictions of the desired horizons. Such a linear mapping of the same features to distinct optimal predictions makes the gradient computation challenging for updating model parameters in the relevant directions. This phenomenon increases the likelihood of model convergence to local optima, particularly in the case of flow data that comprises substantial variations among state values of distinct time steps.

Second, the temporal component in spatial–temporal convolutional layers is mainly designed to capture the shared temporal patterns across all data sources. This shared design makes it insufficient to capture the complex non-linear temporal dependencies within each data series. In fact, a single traffic series exhibits diversified temporal patterns, i.e., the traffic readings of a data source at different time steps are highly different due to time-specific conditions, rendering substantial variations within the series, particularly for the traffic flow than the traffic speed as shown by an example in Fig. 1.

To address the abovementioned issues jointly, we propose a novel feature projection scheme that can readily replace the regular one in the traffic prediction framework of TGNNs. Unlike the regular feature projection layer, which mainly focuses on mapping latent features to scalar predictions, the proposed scheme also serves as an additional pattern modeling phase to capture data source-wise patterns. In addition, the regular feature projection layer directly generates scalar predictions from the unified latent representations, which makes the gradient computation challenging for updating model parameters. In contrast, the proposed scheme effectively uses spatial convolutions to generate vectorized feature maps for the desired horizons and then map them into scalar predictions. These horizon-specific feature maps bridge the gap between the unified latent representation and the scalar predictions and learn interactions between them to guide updates of relevant model parameters.

Specifically, the proposed scheme comprises a feature grouping and a feature aggregation phase. The former phase employs group convolution (Cohen & Welling, 2016) to generate and group data source-wise primary feature maps for the desired horizons and enable weight sharing among them to capture diverse patterns of distinct time steps in a unified latent space. Once the primary feature maps are generated, they are reshaped into a $K \times Q$ feature matrix, where K is the feature-length. The input matrix is then fed to the feature aggregation phase, where Q filters of the same $1 \times Q$ shape are utilized to develop Q horizon-specific feature maps. Each horizon-specific feature map aggregates the primary feature maps of the desired horizons via embedding global relationships among the same dimensional feature entries. These horizon-specific feature maps are then mapped to the corresponding scalar predictions, where each is capable of learning the correlation between the input and the corresponding scalar prediction, allowing an update of weights in the relevant direction when the model is back-propagating. We name the proposed scheme GeneRating and Aggregating Features for Horizons (GRAFH).

It is worth noting that GRAFH aims to enhance the predictive performance of existing TGNNs by enhancing the feature projection scheme. This technique has received limited attention in the literature compared to graph generation and spatial–temporal dependency modeling in TGNNs. In contrast to the regular feature projection scheme, which solely focuses on linear mapping, GRAFH effectively enhances the expressiveness of TGNNs by enriching latent features with data source-wise patterns of distinct time steps. In addition, GRAFH establishes a latent association between timesteps' features and scalar predictions to guide weight updates during model training.

Our specific contributions are summarized as follows:

- We propose an effective feature projection scheme to improve the pattern modeling ability of TGNNs for multi-step traffic forecasting without substantial changes to the core architectures of the remaining modules.
- We make efficacious use of spatial convolutions to generate vectorized feature maps for the desired horizons rather than directly generating scalar predictions, which is a novel contribution of this work. These horizon-specific feature maps discover correlations between input and the corresponding predictions to make them closer to the true values. The proposed scheme is parameter-efficient and improves the robustness of TGNNs.
- We perform multiple experiments on both traffic flow and speed data to demonstrate that the proposed scheme improves the predictive performance of TGNNs. Additionally, we conduct a case study to validate that the performance gain is directly linked to the quality of predictions and the nature of the given data series.

The remainder of this paper is structured in the following sections: Section 2 provides background on the standard traffic forecasting framework. Section 3 defines the traffic forecasting problem. Section 4 presents the proposed scheme and a detailed comparison with the standard mechanism. Section 5 discusses experimental settings and results on traffic prediction, and Section 6 summarizes the writing with concluding remarks and future directions.

2. Background on traffic forecasting framework

TGNNs for traffic prediction are broadly classified into convolutional neural networks (CNNs) and recurrent neural networks (RNNs) based models. Both of these methods generally employ graph neural networks (GNNs) (Veličković et al., 2018; Welling & Kipf, 2016)

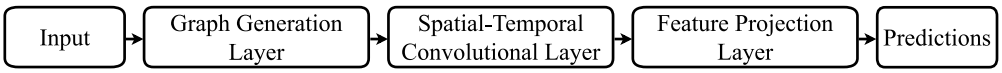

Fig. 2. Standard traffic forecasting framework of TGNNs.

for encoding spatial patterns but differ in terms of temporal correlation modeling. CNNs-based methods exploit variants of standard convolution (LeCun, Bottou, Bengio, & Haffner, 1998) in the temporal component, such as dilated convolution (Wu et al., 2020). RNNs-based methods utilize RNN architectures, such as long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014). Although CNNs have previously shown promising performance, they primarily lack flexibility due to their implicit temporal modeling mechanism, which makes the time steps indiscernible, resulting in a lack of explicit interpretation (Li et al., 2022; Shi et al., 2023). In this work, we focus on RNNs-based methods that benefit from the recurrent operations, explicitly designed for sequential data that naturally fit on modeling temporal correlations in sequential traffic data (Li et al., 2022; Salinas, Flunkert, Gasthaus, & Januschowski, 2020; Tang et al., 2020; Yao et al., 2018). In the following, we provide a detailed overview of TGNNs in terms of the main building blocks, which include a graph generation layer, a spatial–temporal convolutional layer, and a feature projection layer, as illustrated in Fig. 2. The purpose of this categorization is two-fold: (1) to provide a brief background on the operation of the components in the traffic forecasting framework and (2) to highlight the contributions of existing works to these components.

2.1. Graph generation layer

Constructing an adjacency matrix is a primary phase in traffic prediction since the non-Euclidean topological information is typically extracted from the graph adjacency. TGNNs mainly consider three assumptions to design an adjacency matrix: (1) pre-defined, (2) adaptive, and (3) dynamic. For pre-defined adjacency, existing methods (Guo, Lin, et al., 2019; Huang et al., 2021; Li et al., 2018; Wang et al., 2020; Yu et al., 2018; Zheng et al., 2020) utilize distinct distance metrics to compute the graph structure in advance. On the other hand, Wu et al. (2019) introduce the adaptive adjacency scheme that does not require any pre-defined structure and is dynamically learned during model training to discover the hidden spatial relationships among the graph nodes. The latter scheme has recently become an integral component of the modern traffic prediction framework, in which two randomly initialized node embedding matrices are defined, where each row represents an embedding vector of a node. The dimensions of these embedding vectors are considered to be the attributes of the graph nodes. Hence, the spatial dependency between a pair of nodes is inferred via computing the similarity between their randomly initialized attributes. Wu et al. (2020) extends the adaptive scheme to capture asymmetrical relationships between the graph nodes. Bai et al. (2021) assumes that all nodes in a traffic graph belong to the same space and should be represented via a single feature vector whether they appear as a source or target in the given graph. Hence, they define a single node embedding matrix for learning the adaptive adjacency matrix.

Both the pre-defined and adaptive adjacency matrices are static and do not change with time, which limits the ability of the learning model to capture the dynamic characteristics of the traffic network. To handle this problem, Li et al. (2022) utilize a GNN-based hypernetwork architecture to dynamically capture proximity among node embeddings. Lan et al. (2022) extend the idea of dynamic adjacency by capturing spatial relationships among nodes via mining dynamic attributes from the observed traffic values. Shi et al. (2023) utilize the gating mechanism and sparsely connected layer to develop an adjacency update component for constructing the dynamic adjacency matrix. On the other hand, Ye, Xiao, Zhou, Li, Zang, and Zhang (2023) generate multiple graphs to embed diverse contextual correlations among nodes from prior knowledge and then devise a dynamic graph adjustment component to update the adjacency matrix in each training step. Notably, dynamic strategies have also been explored in other related domains to adapt to the evolving nature of real-world networks. For instance, Dong et al. (2023) develop a novel dynamic backhaul topology generation scheme that can adaptively allow changes to the backhaul architecture to capture the different traffic patterns in cellular networks. In addition, Mohajer et al. (2023, 2022) present dynamic optimization models to reduce energy consumption in heterogeneous cellular networks.

2.2. Spatial–temporal convolutional layer

The spatial–temporal convolutional layers in TGNNs involve a graph convolution operation for modeling spatial dependencies and a recurrence operation for modeling temporal dependencies. Existing works on these layers mainly differ in terms of techniques for merging graph convolution with recurrence to model spatial–temporal dependencies simultaneously. For instance, Zhao et al. (2019) replace the linear projections in GRU (Cho et al., 2014) with the standard graph convolution to model spatial–temporal dependencies. Li et al. (2018) replace the standard graph convolution with diffusion graph convolution in GRU units to model spatial–temporal dependencies. Wang et al. (2020) exploit a transformer layer (Vaswani et al., 2017) on the top of the GRU units to additionally embed long-term temporal dependencies. Bai et al. (2021) utilize the learnable node embeddings with a weight pool matrix in graph convolution during the recurrence operation to additionally learn adaptive node parameters. Chen, Segovia, and Gel (2021) incorporate zigzag persistence with time-aware graph convolution in the GRU units to enhance spatial–temporal modeling capabilities. Li et al. (2022) utilize a weighted average sum of graph convolution on the pre-defined and dynamic adjacencies and then replace the matrix multiplication in GRU units with the dynamic graph convolution to capture complex traffic spatial–temporal dependencies. Shi et al. (2023) extend the prior idea by integrating the pre-defined and dynamic adjacency matrices to enable dynamic graph convolution and then integrating the resultant operations in the GRU units. Jiang, Wu, Chen, Zhang, and Kim (2023) further introduce temporal embeddings in the graph convolution and then incorporate the resulting operation in GRU for spatial–temporal dependency modeling. On the other hand, Jiang, Wang, et al. (2023) inject the memorization and distinguishing capabilities into spatial–temporal convolution by utilizing the memory networks to enhance dependency modeling.

2.3. Feature projection layer

In multi-step traffic prediction, the output of the last spatial–temporal convolutional layer is fed to the feature projection layer. In this layer, most existing methods (Li et al., 2022, 2018; Yu et al., 2018) employ a fully-connected layer to generate predictions autoregressively through the number of desired horizons. However, this setting raises inconsistency between training and testing since a model learns to make predictions for one step during training and is expected to produce predictions for multiple steps during inference (Wu et al., 2019). To handle this problem, Wu et al. (2019) introduce a multi-step feature-to-prediction mapping using planner convolution. In this approach, traffic state predictions are directly obtained for the desired number of steps by applying a linear transformation that projects the output of spatial–temporal convolution to real-valued predictions. To

do so, the number of convolutional filters in the feature projection layer is set as a factor of step length to get the desired output dimensions. This approach has been widely adopted in many recent state-of-the-art traffic predictors (Bai et al., 2021; Chen et al., 2021; Jiang, Wu, et al., 2023; Sun et al., 2022; Wu et al., 2020).

In contrast to the previous works, this work does not introduce a new traffic prediction model. Instead, it investigates whether and how the predictive performance of existing TGNNs can be enhanced for multi-step traffic prediction. The majority of the existing studies introduce new TGNNs, with a primary focus either on graph generation to capture various structural information of traffic networks or on the spatial–temporal convolutional layer to model shared patterns across distinct data sources. These efforts are aimed at improving the traffic prediction performance with new architectures. This work, however, takes a different approach to improve the predictive performance by assessing the potential of the feature projection layer in capturing data source-wise patterns and input–output interactions when mapping flow features to predictions.

The main objective of this work is to introduce an innovative and effective feature projection scheme with two fundamental goals. Firstly, it aims to serve as a pattern modeling phase to embed data source-wise patterns in the latent space, thereby enriching latent temporal features with diverse patterns. Secondly, it aims to enhance the feature-to-prediction mapping process by generating vectorized feature maps that learn interactions between latent features and scalar predictions. These feature maps serve as guides for weight updates during back-propagation. It is worth noting that this work does not suggest any substantial changes to the other modules of TGNNs. Instead, the main focus is on the feature projection scheme, which has not yet been adequately explored in previous studies, even though it remains a vital element of TGNNs for traffic forecasting.

3. Preliminary concepts

We begin this section with problem definition and then provide a brief background on some key concepts required to define the proposed method.

3.1. Problem formulation

Multi-step traffic forecasting is typically viewed from a graph perspective. Data sources in a traffic network are primarily regarded as nodes in a graph, which are linked via hidden dependencies. Let $\mathcal{I} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ be a directed, undirected, weighted, or unweighted input graph, representing a traffic network. $\mathcal{V}$ denotes the set of nodes with size $|\mathcal{V}| = N$, where each node can either be a sensor or a road segment/intersection. $\mathcal{E}$ denotes the set of edges, representing the connectivity structure between graph nodes. $\mathbf{A} \in \mathbb{R}^{N\times N}$ is either a binary, geometric, or adaptive adjacency matrix containing topological information of the traffic network.

In this work, we focus on the direct generation of multi-step predictions (Bai et al., 2021; Chen et al., 2021; Wu et al., 2020, 2019). Let $\mathbf{X}_{N,t} = \{x_{1,t}, x_{2,t}, \dots, x_{N,t}\} \in \mathbb{R}^{N\times 1}$ be the recorded traffic state values of N nodes at time step t, then the states of historical data for T time steps can be written as $\mathcal{X} = \{\mathbf{X}_{N,0}, \mathbf{X}_{N,1}, \dots, \mathbf{X}_{N,T}\}$. Given a sequence of historical traffic states for T time steps, the goal is to predict future state values for the next Q steps:

$$\mathcal{P} = \mathcal{F}_\theta(\mathcal{X}, \mathcal{I}), \tag{1a}$$

$$\{\mathbf{P}_{N,t+1}, \mathbf{P}_{N,t+2}, \dots, \mathbf{P}_{N,T+Q}\} = \mathcal{F}_\theta(\mathbf{X}_{N,t}, \mathbf{X}_{N,t-1}, \dots, \mathbf{X}_{N,t-T+1}, \mathcal{I}), \tag{1b}$$

where $\mathcal{F}$ denotes the state forecasting function, θ denotes the learnable model parameters, and $\mathcal{P}$ denotes the set of future state values.

3.2. Spatial–temporal convolution

The spatial–temporal convolution is mainly conducted using a combination of a GCN-based and an RNN-based architecture (Hochreiter & Schmidhuber, 1997; Welling & Kipf, 2016). The GCN-based component is used for spatial dependency modeling, and the RNN-based component is utilized for temporal dependency modeling. GCNs fuse each node's information with its neighbors' information via graph convolution to model the spatial correlations among nodes in the given graph. The nodes' representations obtained from a single graph convolutional layer are defined as follows:

$$\mathbf{X} = \sigma(\hat{\mathbf{A}}\mathbf{X}\mathbf{E}\mathbf{W}), \tag{2}$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with $\mathbf{I}$ denoting the identity matrix for adding self-loops, $\mathbf{E}$ denotes the node embedding matrix, $\mathbf{X}$ denotes the feature matrix, and $\mathbf{W}$ represents the weight matrix for learning model parameters. The GCN representation encodes the immediate neighborhood of each node in the graph, where several GCN layers can also be stacked on top of others as follows:

$$\mathbf{X}^{l+1} = \sigma(\hat{\mathbf{A}}^l\mathbf{X}^l\mathbf{E}^l\mathbf{W}^l), \tag{3}$$

where l denotes the number of a specific GCN layer.

Among RNNs, the temporal component typically employs the best-performing architecture GRU to capture temporal correlations. Mathematically, the GRU operations (Bai et al., 2021; Chen et al., 2021) are defined as follows:

$$\mathbf{Z}_t = \sigma(\hat{\mathbf{A}}[\mathbf{X}_{N,t}, \mathbf{H}_{t-1}]\mathbf{E}\mathbf{W}_Z + \mathbf{E}\mathbf{B}_Z), \tag{4a}$$

$$\mathbf{R}_t = \sigma(\hat{\mathbf{A}}[\mathbf{X}_{N,t}, \mathbf{H}_{t-1}]\mathbf{E}\mathbf{W}_R + \mathbf{E}\mathbf{B}_R), \tag{4b}$$

$$\mathbf{H}'_t = \tanh(\hat{\mathbf{A}}[\mathbf{X}_{N,t}, \mathbf{R}\odot\mathbf{H}_{t-1}]\mathbf{E}\mathbf{W}_{H'} + \mathbf{E}\mathbf{B}_{H'}), \tag{4c}$$

$$\mathbf{H}_t = \mathbf{Z}\odot\mathbf{H}_{t-1} + (1-\mathbf{Z})\odot\mathbf{H}'_t, \tag{4d}$$

where $\mathbf{X}_{N,t}$ and $\mathbf{H}_t$ represent the input and output at time step t. Operators $[,]$ and $\odot$ denote the concatenation and Hadamard product, respectively. $\mathbf{Z}$ and $\mathbf{R}$ are reset and update gates, respectively. $\mathbf{W}_Z, \mathbf{W}_R, \mathbf{W}_{H'}$ are learnable model parameters, and $\mathbf{B}_Z, \mathbf{B}_R, \mathbf{B}_{H'}$ are learnable biases.

3.3. Group convolution

The standard planner convolution is an effective operation for modeling numerous data forms, including text and images. Given an input I and a filter ψ, the standard planner convolution (*) in a neural network is defined as follows:

$$[I * \psi](j) = \sum_{y\in Z^2}\sum_{f=1}^{F} I_f(y)\psi_f(j-y), \tag{5}$$

where F denotes the number of channels and Z^2 is the sampling grid. The above equation implies that the planner convolution is conducted by first shifting the filter by j and then computing the dot product with the input I.

Group convolution (Cohen & Welling, 2016) generalizes planner convolution by generating multiple rotated copies for each kernel via spatial transformations, such as rotation and reflections. These transformations allow convolutional weight-sharing to enhance network expressive capacity without introducing new model parameters. Notably, group convolution is implemented in two ways. The first implementation is based on rotation, while the second includes additional reflections. In this study, we utilize the former, which generates four filters, each by 90-degree counter-clockwise rotation. This operation is defined as follows:

$$[I * \psi](g) = \sum_{y\in Z^2}\sum_{f=1}^{F} I_f(y)\psi_f(g^{-1}-y), \tag{6}$$

where g is a rotation transformation matrix and is discussed in Section 4.3.

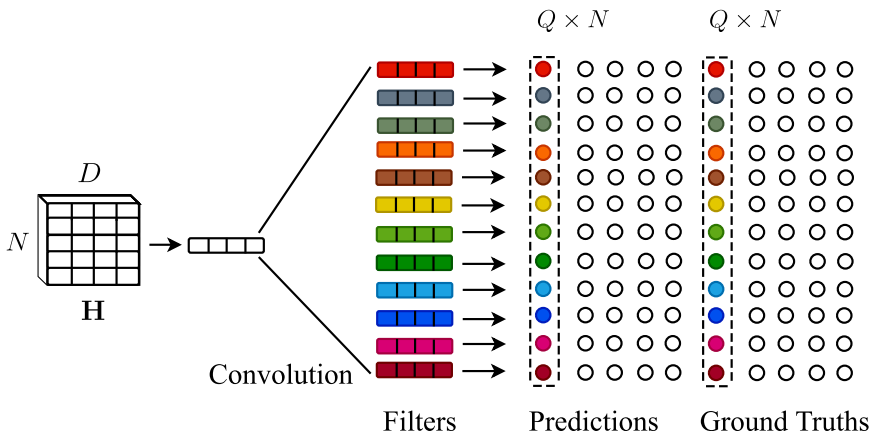

Fig. 3. Architecture of the regular feature projection layer in TGNNs. **H** denotes the output of the last spatial–temporal convolutional layer with N nodes of D dimensions.

4. Methodology

This section introduces the proposed GRAFH and provides a detailed discussion of how it can replace the regular feature projection layer in TGNNs for traffic forecasting. It is noteworthy that the GRAFH does not bring any substantial changes to the core framework of traffic forecasting, except the feature projection scheme, to capture node-wise patterns of distinct time steps and provide horizon-specific weights for guiding gradient computation. We pick four existing state-of-the-art TGNNs (Bai et al., 2021; Chen et al., 2021; Wang et al., 2020; Zhao et al., 2019) and replace their feature projection layer with the proposed GRAFH. In addition, GRAFH maintains the same inputs as implemented in the baselines without introducing any external information. This is important for a fair examination to know whether performance improvement is contributed by the proposed GRAFH.

4.1. Overview of the proposed scheme

As shown in Fig. 3, the linear transformation defined in the regular feature projection layer maps a unified feature representation of the given time steps to predictions of the desired horizons. To this end, a specific filter is defined for each desired horizon to map the unified feature representation to the corresponding prediction. In this case, each representative filter attempts to deduce a correlation between the unified latent representation and the corresponding output. When training the model, it becomes challenging during back-propagation to update weights in the direction of each ground truth since each filter is mapping the same features, making it difficult to render distinct predictions closer to the true ones. This problem is nominal when optimal predictions are expected to lie in a limited range, i.e., when the variance between state values is minor, such as in the case of speed data. However, in the case of data with notable fluctuations, e.g., traffic flow, it becomes challenging to predict distinct values lying at a major distance from others.

The principal objective of GRAFH is to generate feature maps for the desired horizons and then transform those feature maps into predictions rather than directly generating scalar outputs from a single feature map of all given time steps. In fact, GRAFH renders a horizon-specific feature map capable of discovering the interaction between the input and the prediction of the corresponding horizon. In addition, diverse patterns not only exist globally in shared form across all nodes but also distinct time steps of a node exhibit substantial variations, which must be captured during model training. Luckily, we can encode such patterns when generating horizon-specific feature maps rather than bringing any modification to the core forecasting framework.

4.2. Architecture of the proposed scheme

There are two principal stages in GRAFH: grouping features and aggregating features, roughly resembling the encoding-decoding technique. In the former stage, group convolution (Cohen & Welling, 2016) is employed, where the primary feature maps for all horizons are developed and grouped via rotation transformation. Specifically, we define three filters where each filter is rotated four times by 90 degrees counter-clockwise to form three groups of primary feature maps, which are later provided as input to the feature aggregation stage. The main advantage of this component is that it enables weight sharing among primary feature maps, which is useful in modeling diversified patterns exhibited by group members.

In the latter phase, we design an effective convolution operation based on Nguyen et al. (2018) to aggregate the primary features and generate the final horizon-specific feature maps. Specifically, we organize Q filters of $1 \times Q$ size that model feature entries along the same dimensions of the primary feature vectors to aggregate them in shared latent spaces, each corresponding to a specific horizon. Fig. 4 presents the architecture of GRAFH with a detailed demonstration of how the feature grouping and aggregation phases collaborate with each other. Finally, a fully connected layer transforms the final feature maps into predictions of the desired horizons. In more detail, the core stages of GRAFH are presented in the following.

4.3. Feature grouping phase

GRAFH first implements the feature grouping phase. The core of this phase is to receive the output of the spatial–temporal convolutional layer as the input and simultaneously construct and group primary feature maps so that the patterns of distinct time steps of each node can easily be shared in the corresponding groups without introducing any new model parameters. As shown in Figs. 3 and 4, the input to the feature projection layer is a feature matrix $\mathbf{H} \in \mathbb{R}^{N \times D}$. Each row in $\mathbf{H}$ is the D-dimensional feature vector of a graph node. It is noteworthy that $\mathbf{H}$ is obtained from the last spatial–temporal convolutional layer and can be seen as a unified feature representation of all T time steps. It can directly be provided as input to the feature grouping stage, where group convolution is utilized to construct and group primary feature maps:

$$\mathcal{G}_F = \text{GCONV}_\theta(\mathbf{H}), \tag{7}$$

where GCONV represents group convolution, θ denotes learnable weights in the grouping stage, and $\mathcal{G}$ is a feature tensor, representing F output groups of primary feature maps. For instance, group convolution is the generalization of planner convolution, where multiple rotated copies are formed for each kernel by spatial rotation.

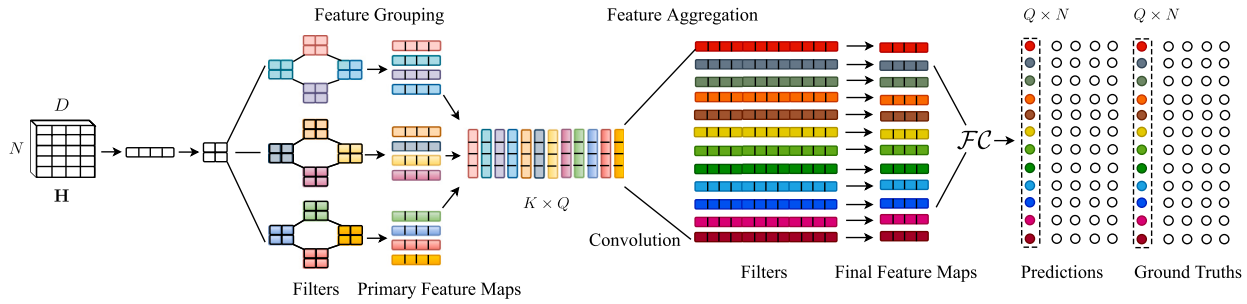

Fig. 4. Architecture of the proposed feature projection scheme. The feature grouping phase generates and groups primary feature maps. The feature aggregation phase aggregates the primary feature maps and generates the final feature maps.

In the case of multi-step prediction for $Q = 12$ desired horizons, we define Q/U filters, where $U = 4$ is the number of spatial rotations. Each filter is then rotated U times by 90 degrees counter-clockwise to generate $F = 3$ groups of filters, each containing U copies, making a total of $FU = Q$ rotated kernels. The number of filter groups is equivalent to the number of feature map groups since each filter group corresponds to a specific group of feature maps. Hence, both quantities are represented by the same letter F. The filter generation process simply makes use of rotation transformation to make specific kernels for all horizons organized in F groups. Each rotated kernel then computes a dot product with feature entries of each row from $\mathbf{H}$ to construct a primary feature map for each desired horizon.

Putting all together, the first filter generates U primary feature maps for the first U horizons (5 min – 20 min), the second filter for the second U horizons (25 min – 40 min), and the third filter for the remaining U horizons (45 min – 60 min). This convolution operation enables a higher degree of convolutional weight sharing among feature maps that improves the pattern handling capability of the feature projection layer without increasing the number of parameters.

The group convolution operation for F groups of primary feature maps ($\mathbf{H} \rightarrow \mathcal{G}_F$) can be written as follows:

$$\mathcal{G}_F = [\mathbf{H} * \psi_F](g), \tag{8a}$$

$$= \sum_{y \in Z^2} \sum_{f=1}^{F} \sum_{u=1}^{U} \mathbf{H}_{f,u}(y) \psi_{f,u}(g^{-1}y), \tag{8b}$$

where Z^2, $*$, and ψ denote the sampling grid, convolution operation, and convolutional filters, respectively. g is the rotation matrix given as follows:

$$g(m,a,b) = \begin{bmatrix} cos(m\pi/2) & -sin(m\pi/2) & a \\ sin(m\pi/2) & cos(m\pi/2) & b \\ 0 & 0 & 1 \end{bmatrix}, \tag{9}$$

where $(a,b) \in Z^2$ and $0 \leq m < U$. Precisely, the group operation on points in Z^2 is implemented through multiplication of the transformation matrix $g(m,a,b)$ in Eq. (9) by the homogeneous coordinate vector $y(a',b')$ of a point (a',b') as follows:

$$gy = \begin{bmatrix} cos(m\pi/2) & -sin(m\pi/2) & a \\ sin(m\pi/2) & cos(m\pi/2) & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a' \\ b' \\ 1 \end{bmatrix}. \tag{10}$$

After the transformation of all points in a filter, it is convolved with $\mathbf{H}$. The same procedure is performed for all U spatially rotated copies of F filters, which returns $F \times U$ feature maps in the output tensor $\mathcal{G}_F \in \mathbb{R}^{N \times F \times U \times D' \times D'}$. $\mathcal{G}_F$ is then reshaped into $\mathcal{G} \in \mathbb{R}^{N \times Q \times K}$ to obtain 1D K-length primary feature maps for $FU = Q$ horizons of each node.

To sum up, the feature grouping stage utilizes group convolution to construct primary feature maps into groups and simultaneously shares weights among group members. Meanwhile, the rotation transformation becomes more efficacious in forming feature maps since the same filter is utilized for all group members. Thus, the diverse patterns of group members can dynamically be learned in a shared latent space.

4.4. Feature aggregation phase

With group convolution in the previous step, we construct primary feature maps so that the model training process learns the inter-dependency between distinct horizons of a node explicitly captured in the filters. Connecting feature maps of distinct horizons renders their corresponding patterns shared. Specifically, the regular feature projection layer uses a separate filter for each desired horizon that lacks in modeling inter-dependencies between the various horizons of a node since it directly maps features to predictions rather than sharing patterns with the other horizons. Hence, we base the feature aggregation phase on the hypothesis that aggregating features of all horizons resembles aggregating diverse patterns from the given data. We conduct this by modeling feature entries of all horizons at the same dimensions.

After feature grouping, the output of the previous phase is then given as input to the feature aggregation phase. In this stage, the input tensor $\mathcal{G} \in \mathbb{R}^{N \times Q \times K}$ is reshaped into $\mathcal{G} \in \mathbb{R}^{N \times K \times Q}$ to compute node-wise final feature maps for the desired horizons. For each node, the primary feature maps of all horizons can be viewed as a $K \times Q$ matrix, which we term $\mathbf{M}$ for mathematical representation. Specifically, the primary feature maps are reshaped from row to column representation, where $\mathbf{M}_{i,:} \in \mathbb{R}^{1 \times Q}$ denotes the ith row of $\mathbf{M}$ containing a single feature entry for all Q horizons.

Let us demonstrate the pattern aggregation process of this phase via a single convolutional filter and a single row of $\mathbf{M}$ in $\mathcal{G}$. Suppose that we define the qth convolutional filter as $\omega_q \in \mathbb{R}^{1 \times Q}$. When operated over ith row of $\mathbf{M}$, ω_q aggregates the corresponding single feature entries of all horizons into a single feature value v_i. In principle, this feature value captures the global relationship between the same dimensional entries, which is one of the important ingredients for capturing diversified patterns in graph data. This process continues for the remaining rows, i.e., ω_q is operated over each row of $\mathbf{M}$ to generate K aggregated feature values that collectively form a final feature map $\mathbf{v}_q$:

$$\mathbf{v}_q = [v_1, v_2, \ldots, v_i, \ldots, v_K]. \tag{11}$$

In addition, each aggregated feature value is computed via the following convolutional operation:

$$v_i = \sigma(\omega \cdot \mathbf{M}_{i,:} + b), \tag{12}$$

where σ denotes the activation function and b denotes the scalar bias.

Generally, we define Q such convolutional filters to generate Q final feature maps for Q horizons, where each filter captures the global associations between the same dimensional feature entries of all Q horizons as follows:

$$\mathcal{G}' = \sum_{n=1}^{N} \sum_{q=1}^{Q} \Big\|_{i=1}^{K} \sigma(\omega_q \cdot \mathbf{M}_{i,:} + b_q), \tag{13}$$

where $\|$ represents the concatenation operation. The above operation returns the feature tensor $\mathcal{G}' \in \mathbb{R}^{N \times Q \times K}$. Then, a fully connected

layer $\mathcal{FC}$ is utilized to transform each K-length feature map to a single-dimensional prediction value of the corresponding horizon:

$$\mathcal{P} = \mathcal{FC}(\mathcal{G}'), \tag{14}$$

where $\mathcal{P} \in \mathbb{R}^{N\times Q\times 1}$ is finally reshaped to prediction matrix $\mathbf{P} \in \mathbb{R}^{Q\times N}$. The matrix $\mathbf{P}$ is then provided to the loss function, where predictions are matched against the ground truths for gradient computation.

4.5. Feature aggregation example

Modeling entries of primary feature maps at the same dimension increases the expressiveness of TGNNs through additional interactions between features of distinct horizons. For example, we consider the case of the regular feature mapping mechanism. Here, a K-dimensional feature vector of a node is convolved with a K-dimensional filter:

$$\begin{bmatrix} a_1 & a_2 & a_3 & \dots & a_K \end{bmatrix} * \begin{bmatrix} z_1 & z_2 & z_3 & \dots & z_K \end{bmatrix}, \tag{15}$$

where a and z represent entries of the feature vector and filter, respectively. The above regular convolution operation models interactions within the given feature vector and produces the scalar output. Instead, we generate the primary feature maps and then compose them into a matrix. By doing so, we obtain the following representation for the convolution operation that enables us to model interactions between the feature entries of all horizons:

$$\begin{bmatrix} a_1 & b_1 & c_1 & \dots & l_1 \\ a_2 & b_2 & c_2 & \dots & l_2 \\ a_3 & b_3 & c_3 & \dots & l_3 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_K & b_K & c_K & \dots & l_K \end{bmatrix} * \begin{bmatrix} z_1 & z_2 & z_3 & \dots & z_Q \end{bmatrix}, \tag{16}$$

where letters from a to l represent feature entries of the $Q = 12$ horizons. This convolution encapsulates entries at the same dimensions from all primary feature maps into a length-K final feature map, encoding many same dimensional characteristics, including the intrinsic spatial relationships between features of distinct horizons. Hence, replacing the scalar output with a vector output accounts for capturing diverse patterns from one horizon to another.

4.6. Training configurations

For training TGNNs with GRAFH, we use the following standard settings reported in the literature (Bai et al., 2021; Wu et al., 2020, 2019). We split the historical data $\mathcal{X}$ into training, validation, and test sets. We then slide a window of size $T + Q$ over the training set to generate training sequences. For each training sequence, the first T elements are used as the input, while the remaining Q elements are used as the ground truths. Both ground truths $\mathbf{X}$ and predictions $\mathbf{P}$ for Q elements are provided to the following L1 loss function $\mathcal{L}$, which is then optimized via Adam optimizer (Kingma & Ba, 2015).

$$\mathcal{L}_\Theta(\mathbf{X}, \mathbf{P}) = \sum_{q=1}^{Q} |\mathbf{X}_{N,q} - \mathbf{P}_{N,q}|, \tag{17}$$

where N represents the number of nodes, q denotes the qth horizon, and Θ symbolizes all learnable parameters.

5. Experiments

We conduct a series of comprehensive experiments on two real-world traffic datasets to demonstrate the performance of the proposed scheme with existing TGNNs. Initially, we examine the performance gain over the regular mapping mechanism of multi-step flow prediction. Then, we conduct a case study demonstrating that GRAFH produces optimal predictions and reduces errors on most nodes. Moreover, we carry out ablation experiments on GRAFH architecture to highlight the contribution of each component. Further, we perform a hyper-parameter test to demonstrate the robustness of the proposed scheme. In addition, we analyze how GRAFH reduces computation costs by utilizing light yet effective convolutional operations compared to the existing scheme. Finally, we test the generalization property of GRAFH by performing experiments on the speed data.

5.1. Datasets details

In order to assess the performance of the proposed scheme, we employ two real-world standard traffic datasets: PeMSD4 and PeMSD8. PeMS stands for Caltrans Performance Measure System (PeMS), which collects the real-time highway traffic of California every 30 s. The PeMSD4 dataset contains the traffic flow and speed data in the San Francisco Bay Area collected via 307 loop detectors during the period 1/Jan/2018 – 28/Feb/2018. The PeMSD8 dataset comprises traffic flow and speed data recorded via 170 loop detectors in the San Bernardino area from 1/Jul/2016 to 31/Aug/2016. Both datasets are aggregated into 5-minute windows, resulting in 288 data points per day. We closely follow the experimental settings reported in Bai et al. (2021) that include standard data normalization and filling missing values via linear interpolation. Following the standard settings of traffic forecasting, the 12 steps historical data is composed as the input, and the following 12 steps data is provided as the output, i.e., one-hour historical data is utilized to forecast the next hour's data. The training, validation, and test sets are formed by splitting the reported datasets with a ratio of 6:2:2.

5.2. Evaluation protocols and hyper-parameters

The performance of traffic prediction methods is evaluated using Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE), defined as follows:

$$\text{MAE}(\mathbf{X}, \mathbf{P}) = \frac{1}{Q} \sum_{q\in Q} |\mathbf{X}_{N,q} - \mathbf{P}_{N,q}|, \tag{18}$$

$$\text{RMSE}(\mathbf{X}, \mathbf{P}) = \sqrt{\frac{1}{Q} \sum_{q\in Q} (\mathbf{X}_{N,q} - \mathbf{P}_{N,q})^2}, \tag{19}$$

$$\text{MAPE}(\mathbf{X}, \mathbf{P}) = \frac{1}{Q} \sum_{q\in Q} \left| \frac{\mathbf{X}_{N,q} - \mathbf{P}_{N,q}}{\mathbf{X}_{N,q}} \right|. \tag{20}$$

We conduct random search for optimal sets of hyper-parameters on hidden size (D), embedding size (C), batch size (B), and learning rate (η). We found the following sets of hyper-parameters best for the reported datasets: $\{\text{PeMSD4} : D = 36/60/64, C = 10, B = 64, \eta = 0.0001 - 0.003\}$ and $\{\text{PeMSD8} : D = 60/64, C = 2/10, B = 8/64, \eta = 0.0001 - 0.005\}$. We train models with hyper-parameters from these sets for 100/350 epochs on both datasets.

5.3. Baselines

We employ the proposed GRAFH on two testing datasets to predict traffic states. As GRAFH maps latent features of time steps to predictions of horizons, we adopt the following state-of-the-art TGNNs and examine the performance gain caused by incorporating GRAFH:

- T-GCN (Zhao et al., 2019): Temporal graph convolutional network (T-GCN) conjugates GCN with GRU to model the spatial–temporal correlation among traffic series.
- STGNN (Wang et al., 2020): Spatial temporal graph neural network (STGNN) integrates a GNN with GRU and transformer for traffic state forecasting.
- AGCRN (Bai et al., 2021): Adaptive graph convolutional network (AGCRN) merges adaptive graph generation scheme with graph convolution aiming at learning node adaptive parameters and then combines the resultant architecture with GRU for traffic forecasting.

Table 1
Flow prediction results on PeMSD4.

PeMSD4	Horizon 3			Horizon 6			Horizon 9			Horizon 12		
Model	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
T-GCN	20.56	32.13	14.3770	22.16	34.53	15.2757	23.99	37.05	16.6420	26.45	40.32	18.4583
T-GCN + GRAFH (COM)	20.35	31.83	14.1352	**21.86**	34.10	15.0179	**23.70**	36.59	16.3494	**26.18**	39.89	18.2628
T-GCN + GRAFH (DEF)	**20.31**	**31.67**	**13.9640**	21.90	**34.02**	**14.8403**	23.71	**36.48**	**16.1314**	26.20	**39.81**	**17.8564**
STGNN	20.64	32.61	14.0499	23.16	36.30	15.6793	25.73	39.93	17.4155	28.64	44.00	19.4552
STGNN + GRAFH (COM)	20.27	32.16	13.3635	22.92	36.03	15.1055	25.45	39.63	16.9557	28.48	43.78	19.0313
STGNN + GRAFH (DEF)	**20.14**	**31.98**	**13.2225**	**22.66**	**35.68**	**14.9346**	**25.22**	**39.33**	**16.8419**	**28.04**	**43.26**	**18.9191**
Z-GCNETs	18.58	30.20	12.5034	19.42	31.72	**12.9293**	20.10	32.97	13.2830	21.11	34.45	**13.8680**
Z-GCNETs + GRAFH (COM)	18.56	**29.89**	12.3929	19.47	**31.42**	13.0279	20.18	**32.60**	13.4284	21.27	**34.15**	14.2030
Z-GCNETs + GRAFH (DEF)	**18.54**	29.94	**12.3734**	**19.41**	31.46	12.9551	**20.07**	32.61	**13.2719**	**21.09**	**34.15**	13.9284
AGCRN	18.90	30.90	12.5359	19.63	32.17	12.9260	20.15	33.20	**13.1619**	21.14	34.63	**13.8675**
AGCRN + GRAFH (COM)	18.50	29.83	12.3506	19.38	31.31	**12.8877**	20.04	32.48	13.3029	21.09	34.00	14.0266
AGCRN + GRAFH (DEF)	**18.42**	**29.68**	**12.3351**	**19.33**	**31.15**	12.9002	**19.96**	**32.21**	13.3178	**20.90**	**33.60**	13.9478

- Z-GCNETs (Chen et al., 2021): Time-aware zigzag graph convolutional networks (Z-GCNETs) incorporates zigzag persistence into time-aware graph convolutional networks and then combines the resultant architecture with GRU for traffic forecasting.

AGCRN (Bai et al., 2021) and Z-GCNETs (Chen et al., 2021) are evaluated using their publically available source code. For T-GCN and STGNN, we utilize their publically available source code and implement them with the multi-step settings of traffic forecasting in the code-base of AGCRN (Bai et al., 2021). Besides, the following two schemes are designed for evaluating GRAFH.

- DEF: It refers to the default setting of generating primary feature maps for all 12 horizons via setting the output channel of the group convolutional layer to $F = 3$, where each channel has $U = 4$ orientation channels.
- COM: It refers to the compact setting that generates four primary feature maps via setting the number of output channels to $F = 1$, which also has $U = 4$ orientation channels. In such settings, only the final feature maps are horizon-specific but not the primary ones. Also, it allows us to further reduce the number of model parameters.

We tag the given baselines with the "Predictor + GRAFH (SCHEME)" naming pattern. For example, "AGCRN + GRAFH (DEF)" refers to replacing the regular feature projection scheme of AGCRN with GRAFH having default settings. We follow the same naming patterns for all the given baselines.

5.4. Results on traffic flow prediction

Tables 1 and 2 report the traffic flow prediction results of the baselines with both the regular feature projection layer and the proposed GRAFH and compare them via standard evaluation metrics MAE, RMSE, and MAPE. The best scores are highlighted in bold. The statistical assessment of reported scores reveals that a feature projection technique considering shared patterns to generate horizon-specific feature maps like GRAFH can notably enhance the predictive performance of traffic predictors with trivial additional efforts. Several interesting observations from the reported scores can be developed, which are summarized in the following:

- When mapping flow features to predictions, GRAFH outperforms the regular counterpart on all four predictors across the given datasets. In particular, an average 1.1%, 2.2%, 0.1%, and 1.5% MAE reduction and an average 1.4%, 1.7%, 0.94%, and 3.3% RMSE reduction can be observed by replacing the regular feature mapping mechanism with the one implemented by GRAFH (DEF) across PeMSD4 datasets for T-GCN, STGNN, Z-GCNETs, and AGCRN, respectively. On the PeMSD8 dataset, a more significant improvement can be observed where GRAFH (DEF) reduces average MAE and RMSE values of T-GCN by 3.2% and 2.5%, STGNN by 2.2% and 2.1%, Z-GCNETs by 0.44% and 0.34%, and AGCRN by 2.6% and 2.9%, respectively. In addition, it can also be observed that GRAFH (DEF) improves the predictive performance of baselines on both short and long intervals. For example, for AGCRN on PeMSD4, GRAFH (DEF) reduces MAE and RMSE values by 2.54% and 3.95% on the short interval 3, while 1.13% and 2.97% on the long interval 12, respectively. In contrast, on PeMSD8, GRAFH (DEF) reduces MAE and RMSE by 2.54% and 2.9% on the short interval 3, while 2.56% and 3.12% on the long interval 12. These quantitative observations demonstrate the significance of the proposed method in both short and long range predictions depending upon the nature of the dataset. This performance improvement reveals that the proposed GRAFH (DEF) better synchronizes with TGNNs compared to the existing counterpart. Hence, it can be used as a drop-in replacement for the regular feature projection scheme in the traffic forecasting framework of TGNNs.
- The comparison suggests that both GRAFH settings: default and compact, improve performance, indicating that aggregating patterns in a shared latent space are essential, whether among all given time steps or the specific ones. In addition, modeling entries at the same dimensions of feature maps is an additional valuable approach to aggregating the diverse patterns. Both feature grouping and aggregating help TGNNs embed the additional node-wise diverse patterns of distinct time steps in the latent space. The individual contribution of each phase is discussed in the ablation study in Section 5.6.
- GRAFH maintains performance gains for different adjacency generation schemes, the adaptive in the AGCRN, STGNN, the predefined in the T-GCN, and the combination of both in Z-GCNETs. This observation is of significant importance since existing works mainly consider adjacency as the primary variable of predictive performance. Thus, performance gain via GRAFH would be an additional benefit to future works on the other modules of the traffic forecasting framework.
- While GRAFH consistently reduces the prediction errors, the average magnitude is different across the two datasets. In particular, PeMSD4 and PeMSD8 achieve an average 1.2% and 2.1% MAE reductions for all baselines with default GRAFH settings. Such a difference in the performance gains can be linked directly to the nature and size of the datasets since GRAFH improves performance by generating horizon-specific feature maps from the given data without considering any other variable.
- The spatial–temporal convolutional layer of all the reported baselines comprises a GCN and GRU. Thus, they all share a common approach to modeling spatial–temporal dependencies. However, the coordination between both components is different across all the baselines. In addition, STGNN implements an additional

Table 2
Flow prediction results on PeMSD8.

PeMSD8	Horizon 3			Horizon 6			Horizon 9			Horizon 12		
Model	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
T-GCN	16.99	25.69	11.8641	18.00	27.52	12.3454	19.30	29.45	13.1113	21.14	32.10	14.4705
T-GCN + GRAFH (COM)	16.39	25.03	**11.1785**	17.68	27.23	**11.7429**	18.93	29.05	**12.5537**	20.86	31.70	14.0210
T-GCN + GRAFH (DEF)	**16.32**	**24.89**	11.3857	**17.50**	**26.95**	11.8624	**18.70**	**28.78**	12.5583	**20.49**	**31.31**	**13.7051**
STGNN	16.32	25.63	10.2847	18.36	28.93	11.5617	20.22	31.73	12.8276	22.53	35.16	14.3237
STGNN + GRAFH (COM)	15.79	24.78	**9.9238**	17.86	28.26	**11.1605**	19.91	31.39	**12.4178**	**22.16**	**34.72**	**13.6715**
STGNN + GRAFH (DEF)	**15.74**	**24.67**	10.1342	**17.83**	**28.21**	11.3727	**19.86**	**31.26**	12.7336	22.34	34.98	14.2539
Z-GCNETs	14.95	23.54	9.5683	15.92	25.25	**10.0936**	16.54	26.32	10.4185	17.57	27.88	**10.9136**
Z-GCNETs + GRAFH (COM)	15.00	23.59	9.6894	16.06	25.41	10.2049	16.66	26.44	10.5408	17.75	28.10	11.2093
Z-GCNETs + GRAFH (DEF)	**14.84**	**23.38**	**9.5375**	**15.89**	**25.23**	10.0978	**16.43**	**26.19**	**10.4079**	**17.54**	**27.85**	11.0251
AGCRN	14.95	23.59	9.6636	15.98	25.40	10.2129	16.83	26.79	10.725	17.99	28.52	11.4331
AGCRN + GRAFH (COM)	14.87	23.23	9.4409	16.00	25.21	10.1171	16.76	26.36	10.6372	18.07	28.29	11.5597
AGCRN + GRAFH (DEF)	**14.57**	**22.90**	**9.4314**	**15.68**	**24.84**	**10.0743**	**16.27**	**25.87**	**10.6226**	**17.53**	**27.63**	**11.4062**

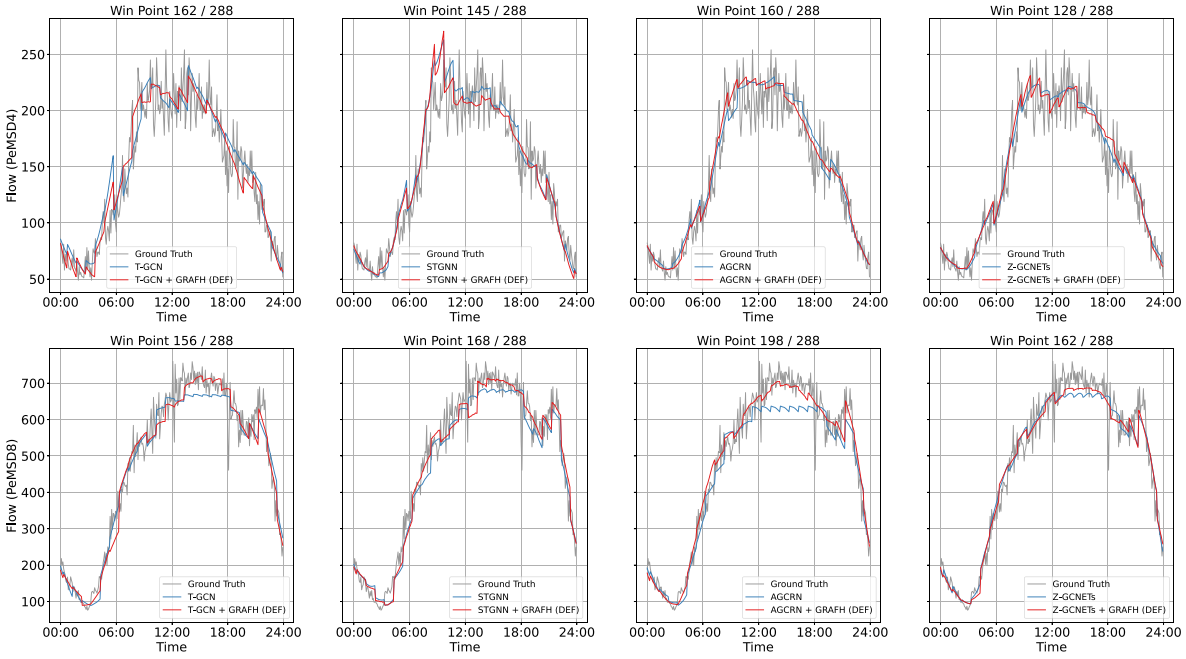

Fig. 5. Predictions vs. ground truths on PeMSD4 and PeMSD8 datasets.

transformer-based component that interacts with GRU to generate predictions. Given different architectural scenarios, GRAFH still proves to be effective, making it a prominent candidate for future works on traffic forecasting.

5.5. Case study

In this section, we conduct a case study to demonstrate how GRAFH improves the predictive performance of TGNNs. Specifically, we conduct the following two experiments:

- We pick node no. 261 and node no. 9 with flow values from the PeMSD4 and PeMSD8 datasets, respectively. The former node has a lower fluctuation range, while the latter has a higher one. We then generate their one-day ahead flow predictions for all baselines with the regular feature projection layer and with the default settings of GRAFH and visualize them against the ground truths in Fig. 5. This visual comparison demonstrates that GRAFH enhances the predictive performance of TGNNs by generating predictions closer to the ground truths. In particular, we define a metric called Win Point, which computes the number of optimal predictions from one-day ahead forecasted values. Optimal predictions refer to those which are the closest to the ground truths. On node 261, AGCRN makes 128 optimal predictions, which GRAFH increases to 160, improving the Win percentage from 44.4% to 55.6%. A similar performance gain can be observed for T-GCN, where the Win percentage is enhanced from 43.8% to 56.3%. However, in the case of STGNN and Z-GCNETs, the Win percentage of GRAFH is almost similar to or lower than the existing counterparts, even though the results are improved on the other metrics reported in Table 1. This suggests that GRAFH improves prediction quality slightly but consistently across most of the nodes in the traffic network. On node 9, a more considerable performance gain can be observed, particularly in the case of AGCRN, for which GRAFH significantly increases optimal predictions from 90 to 198, boosting the Win percentage from 31.25% to 68.76%. A substantial portion of this refinement can be visually observed in Fig. 5 for the peak time period 12:00-18:00, where GRAFH brings the predictions of all reported baselines closer enough to the ground truths.
- We generate a node-to-node RMSE comparison for all baselines with the regular feature projection layer and with the default settings of GRAFH on both testing datasets and visualize them in Fig. 6. The blue dots corresponding to all baselines are clearly visible across most of the nodes in both testing datasets. This

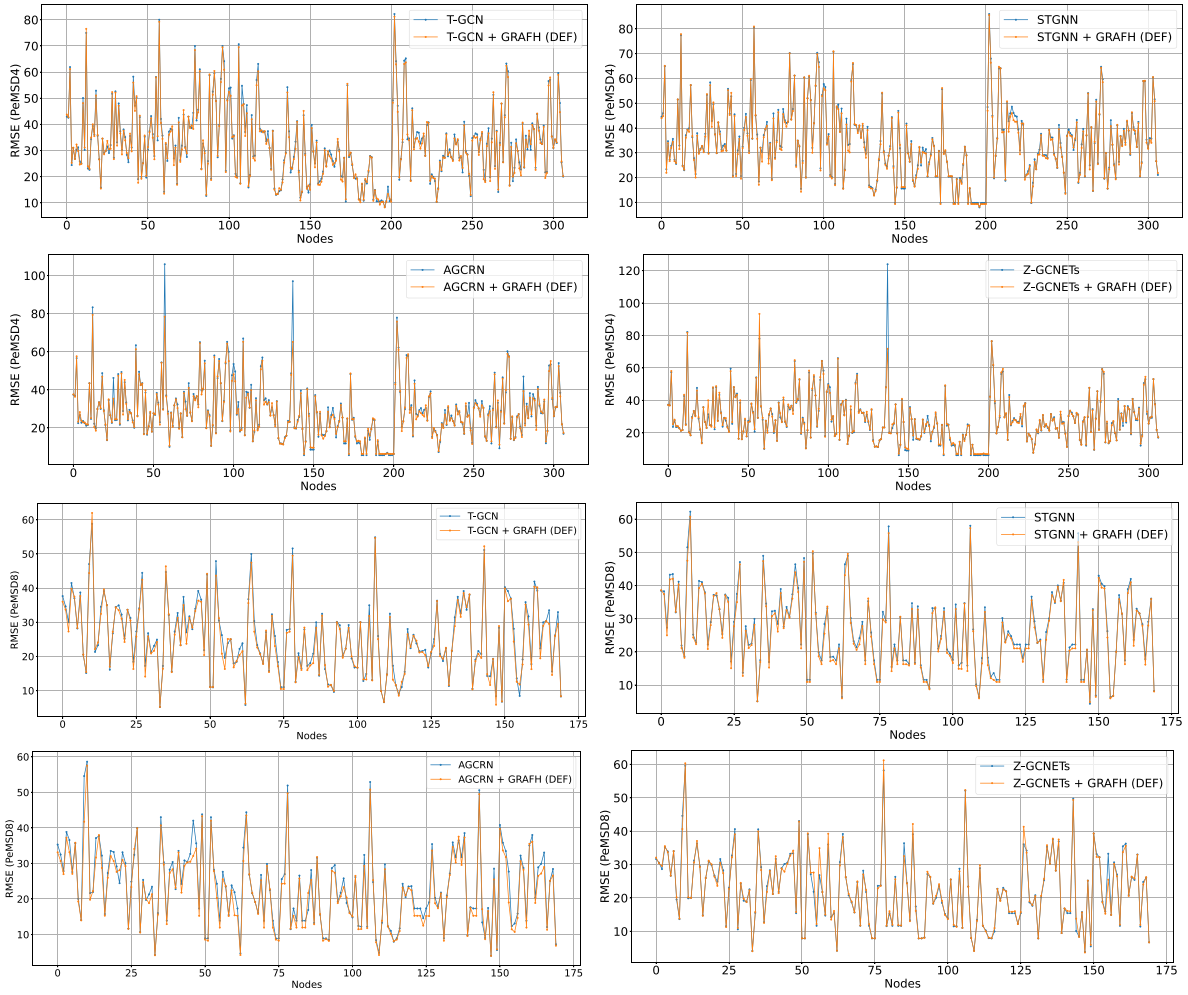

Fig. 6. Node-to-node RMSE comparison on PeMSD4 and PeMSD8 datasets.

observation reveals that GRAFH reduces prediction errors for most of the nodes and thus gains a significant advantage over the existing counterpart.

5.6. Ablation study

Fig. 7 reports the results of the ablation experiments on the flow data of the PeMSD4 and PeMSD8 datasets. Specifically, we design the following two variants of GRAFH (DEF) to evaluate the performance of both components:

- w/o aggregation: In this ablation experiment, the feature grouping phase is retained, wherein the feature aggregation phase, the same dimensional convolution operation, is replaced with a planner 1D convolution which only transforms primary feature maps but does not aggregate them.
- w/o grouping: In the second ablation experiment, the feature aggregation phase is retained. In the feature grouping phase, the group convolution is replaced with planner convolution, where the number of output channels is set to $Q = 12$ to generate Q primary features. Although Q primary feature maps are generated in such settings, they do not share weights and hence not the patterns.

Both the ablated versions are tested with AGCRN. A considerable drop in the performance of both ablated architectures can be observed on both datasets, even worse than the existing one. For example, GRAFH (w/o aggregation) increases the MAE values by 5.8% and 4.2% on PeMSD4 and PeMSD8 datasets, respectively. On the other hand, GRAFH (w/o grouping) increases the MAE values by a large magnitude of 26% and 38% on PeMSD4 and PeMSD8 datasets, respectively. Such a large drop in the performance can be attributed to the feature repetition in the feature generation step, which we avoid by using the grouping and weight sharing operations. These ablation experiments validate that the performance gain of GRAFH is based explicitly on weight sharing and aggregation under the proposed scheme.

In addition, we conduct two paired t-tests: (1) with GRAFH (w/o grouping) and GRAFH (DEF) and (2) with GRAFH (w/o aggregation) and GRAFH (DEF). The paired t-test is defined as follows:

$$t = \frac{m'}{s_m/\sqrt{n}} \tag{21}$$

where n is the number of paired values, m' is the mean, s_m is the standard deviation, and t is the output t-value of the test. Both m' and s_m are computed from the differences between the given paired values. In both these statistical tests, we use the confidence level $\alpha = 0.05$ with MAE values of all horizons. We get the t-values 10.65 and 4.81 for the first and second tests, respectively. We then compare these t-values with the critical t-value 2.201. This critical t-value is obtained from the t-distribution table. As both the computed t-values are greater than the critical t-value, we can safely reject the null hypothesis in both cases that model predictions are equivalent. These tests reveal a significant statistical difference between the proposed method and each ablated version in terms of the MAE values.

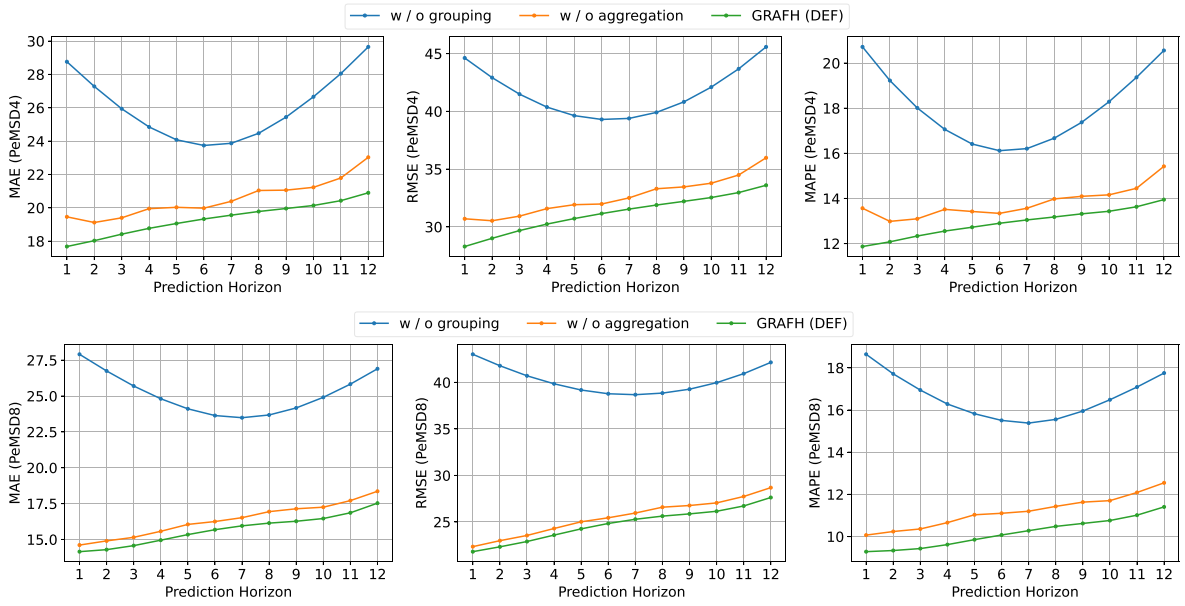

Fig. 7. Results of ablation study on PeMSD4 and PeMSD8 datasets.

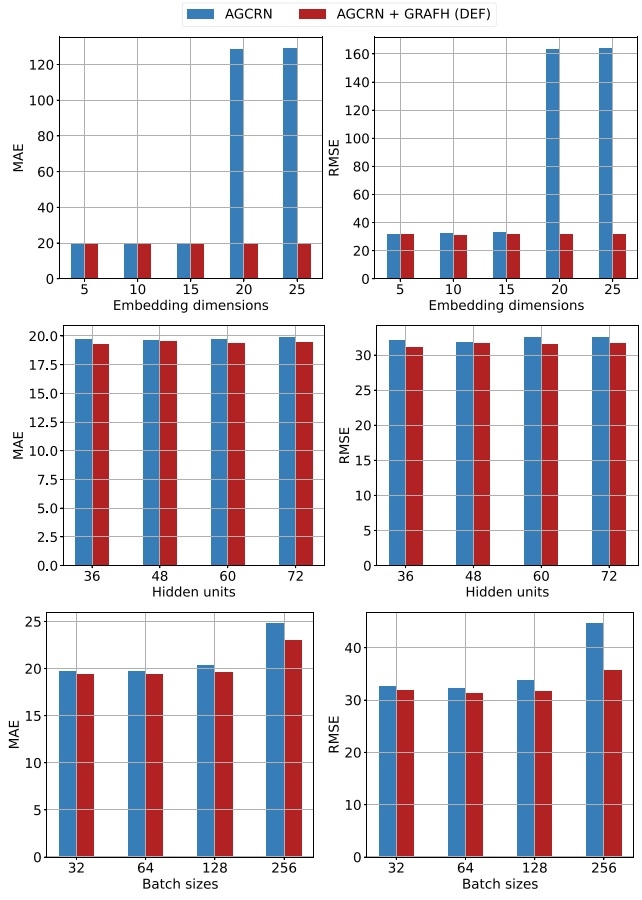

Fig. 8. Comparison on different embedding dimensions, hidden units, and batch sizes.

5.7. Hyper-parameter study

In this section, we evaluate how GRAFH responds to changes in the key hyper-parameters. We conducted experiments with varying batch sizes, hidden units, embedding dimensions, and learning rates to assess their influence on predictive performance. The experiments are performed on AGCRN with both GRAFH (DEF) and the regular feature projection layer on the PeMSD4 dataset with embedding dimensions $C \in \{5, 10, 15, 20, 25\}$, hidden units $D \in \{32, 64, 128, 256\}$, batch size $B \in \{32, 64, 128, 256\}$, and learning rates $\eta \in \{0.0003, 0.0005, \ldots, 0.003\}$. From Fig. 8, it can be observed that the performance of GRAFH (DEF) is nearly stable against both small (5/10) and large embedding dimensions compared to the regular counterpart, which changes drastically for large dimensions (20/25) mainly due to over-fitting that GRAFH typically avoids. In addition, the hidden units have a slightly lower effect on GRAFH performance. For example, the average MAE value for the regular feature projection layer increases by 1% from 36 to 72 hidden units while that for GRAFH increases by a slightly lower margin of 0.82%. Furthermore, both GRAFH and its regular counterpart are not highly sensitive to small and average batch sizes (32/64/128). However, for the large batch size of 256, the average RMSE value of the regular feature projection layer increases by a large margin of 25% compared to the 11% increase in GRAFH's RMSE.

Besides, an interesting observation is on the trade-off between learning rate and predictive performance, shown in Fig. 9. The regular feature projection layer typically requires a precise learning rate to attain considerable convergence, e.g., 0.003 for AGCRN on the PeMSD4 dataset. The proposed GRAFH, however, improves the model's learning ability by deploying horizon-specific weights to be updated in the direction of the corresponding ground truths with varying learning rates while still attaining nearly the same convergence. Hence, GRAFH enhances model robustness and exhibits a nominal sensitivity to hyperparameters compared to the regular counterpart.

5.8. Computation cost

In order to assess the feasibility of GRAFH, we compare the training time and the number of model parameters of the reported baselines with the regular feature projection layer and with default and compact settings of the proposed GRAFH on the PeMSD8 dataset in Table 3. It is noteworthy that all the hyper-parameters, such as the size of node embeddings and hidden layers, are kept constant for a fair comparison. Table 3 shows that both default and compact settings of GRAFH slightly reduce the number of model parameters. For example, in the case of T-GCN, the default and compact settings reduce model parameters by

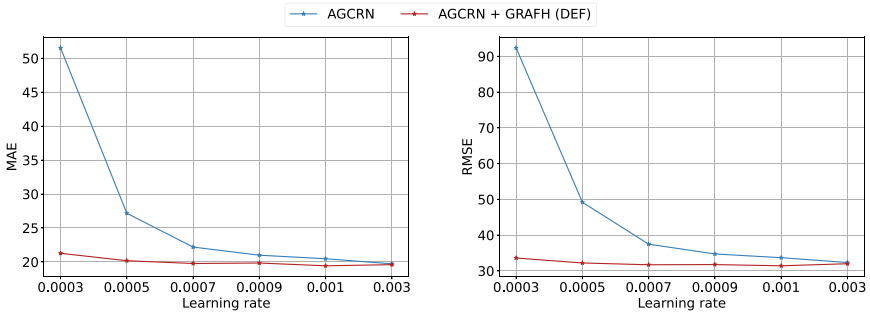

Fig. 9. Comparison on different learning rates.

Table 3
Computation cost.

Model	# of parameters	Training time (epoch)
T-GCN	11.89 K	6.1 s
T-GCN + GRAFH (COM)	11.26 K	6.2 s
T-GCN + GRAFH (DEF)	11.38 K	6.6 s
STGNN	544.33 K	17.3 s
STGNN + GRAFH (COM)	543.70 K	18.8 s
STGNN + GRAFH (DEF)	543.82 K	19.2 s
AGCRN	132.11 K	19.1 s
AGCRN + GRAFH (COM)	131.48 K	19.8 s
AGCRN + GRAFH (DEF)	131.60 K	20.2 s
Z-GCNETs	96.66 K	46.5 s
Z-GCNETs + GRAFH (COM)	96.03 K	47.5 s
Z-GCNETs + GRAFH (DEF)	96.14 K	48.5 s

4.3% and 5.3%, respectively. Thus, saving a fraction of the required memory attributed to the structure of the utilized convolutional operations. Regarding the training time, all baselines with GRAFH run 1–2 s per epoch slower than with the regular counterpart. However, such a slight delay per run is negligible, considering the overall improvement in performance and the reduction in model parameters. Hence, GRAFH is an effective and parameter-efficient feature projection scheme, which can easily be integrated with TGNNs for performance improvement with little additional training time.

5.9. Generalization to speed data

The proposed GRAFH is specifically designed to generate horizon-specific feature maps that capture diverse patterns within each traffic series to enhance the predictive performance of traffic predictors. To validate the generalization property of GRAFH, we subsequently conduct experiments on speed prediction. As discussed earlier, traffic speed data differs from traffic flow data in terms of local patterns, i.e., traffic speed data has minor variations and consistent periodicity than traffic flow data. Tables 4 and 5 report results of traffic speed prediction on PeMSD4 and PeMSD8 datasets. Similar to the previous experiments on traffic flow prediction, it can be observed that the proposed GRAFH also enhances predictive performance on traffic speed data. However, the performance gain is considerably lower than that on the traffic flow prediction. For instance, GRAFH (DEF) with AGCRN reduces the average RMSE by 2.8% on the speed data of the PeMSD4 dataset, which is lower than the 3.3% reduction on the flow data of the same dataset. Similarly, on the PeMSD8 dataset, the average RMSE is reduced by a margin of 2.4% on the speed data compared to the 2.9% reduction on the flow data. This performance gap can be attributed to the variation difference between the speed and flow data. In addition, it can also be observed that T-GCN and STGNN with the regular feature projection layer outperform the proposed GRAFH for horizon 3 on the PeMSD8 dataset. This observation can be justified by the fact that the prediction task is a little easier for small intervals while becoming difficult as the prediction interval increases.

6. Conclusions

In this paper, we propose a novel feature projection scheme for RNNs-based traffic predictors to additionally learn the diverse temporal patterns from each traffic series when projecting features to predictions. Specifically, this work is based on the fact that the traffic data, particularly the flow, comprises varying periods and diverse temporal patterns within each series, which existing predictors may only partially capture since they are biased to the shared patterns across all series. The proposed GRAFH can capture such patterns since it conducts convolutional operations for each series rather than for each temporal period.

By employing group convolution, the temporal patterns of distinct time steps for each node are encoded in a shared latent space while constructing the primary feature maps. Subsequently, an effective technique of 1D convolution is designed to model feature entries of given horizons along the same dimensions to aggregate patterns into the final feature maps. These horizon-specific feature maps play a crucial role in learning interactions between latent features and scalar predictions, enabling the learning model to update weights accurately in the direction of the corresponding ground truths. This has been empirically validated by detailed experimentation.

To evaluate the efficacy of the proposed GRAFH, a series of detailed experiments are carried out on two real-world traffic datasets comprising flow and speed data. The results demonstrate that the proposed GRAFH outperforms the regular counterpart by a significant margin with a nominal time overhead. In addition, the proposed scheme is parameter-efficient and enhances model robustness, which is verified by a detailed hyper-parameter sensitivity test. In the future, we intend to extend the proposed scheme to the auto-regressive settings of multi-step traffic prediction to alleviate the error accumulation problem. Besides, we plan to expand the proposed method to other time-series forecasting domains, including finance, economics, climate, and healthcare.

CRediT authorship contribution statement

Adnan Zeb: Conceptualization, Investigation, Methodology, Writing – original draft. **Shiyao Zhang:** Review & editing. **Xuetao Wei:** Review & editing. **James Jianqiao Yu:** Review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Table 4
Speed prediction results on PeMSD4.

PeMSD4	Horizon 3			Horizon 6			Horizon 9			Horizon 12		
Model	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
T-GCN	1.48	3.06	**2.9530**	1.85	4.09	3.8998	2.12	4.77	4.6459	**2.34**	5.35	5.2900
T-GCN + GRAFH (COM)	1.48	3.08	3.0339	**1.82**	4.05	**3.8378**	**2.09**	4.77	**4.5664**	**2.34**	5.38	**5.2344**
T-GCN + GRAFH (DEF)	**1.47**	**3.05**	3.0122	**1.82**	**4.04**	3.8603	2.10	**4.75**	4.6017	2.35	**5.34**	5.2767
STGNN	**1.41**	**3.01**	**2.7623**	**1.81**	4.12	**3.8222**	2.11	**4.85**	**4.6500**	2.35	**5.40**	5.3217
STGNN + GRAFH (COM)	**1.41**	**3.01**	2.7787	**1.81**	**4.11**	3.8557	**2.10**	**4.85**	4.6863	**2.34**	5.42	**5.3101**
STGNN + GRAFH (DEF)	**1.41**	**3.01**	2.8074	**1.81**	4.12	3.8644	2.11	4.86	4.6762	2.35	**5.40**	5.3355
Z-GCNETs	**1.37**	2.91	2.7988	**1.66**	**3.74**	3.5776	**1.83**	**4.22**	4.0288	**1.97**	**4.56**	4.3866
Z-GCNETs + GRAFH (COM)	1.39	2.94	2.7996	1.67	3.76	**3.5241**	1.84	4.26	**3.9757**	2.00	4.68	**4.3847**
Z-GCNETs + GRAFH (DEF)	**1.37**	**2.89**	**2.7578**	1.67	**3.74**	3.5591	1.86	4.26	4.0579	2.02	4.66	4.4609
AGCRN	1.43	3.08	2.9556	1.71	3.86	3.6374	1.86	4.38	4.0944	2.05	4.78	4.5121
AGCRN + GRAFH (COM)	**1.37**	2.90	**2.7893**	**1.67**	**3.75**	**3.5941**	**1.84**	**4.25**	**4.0491**	2.02	**4.68**	4.4690
AGCRN + GRAFH (DEF)	1.38	**2.89**	2.7980	1.68	3.78	3.6069	1.85	4.30	4.0708	**2.01**	**4.68**	**4.4543**

Table 5
Speed prediction results on PeMSD8.

PeMSD8	Horizon 3			Horizon 6			Horizon 9			Horizon 12		
Model	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
T-GCN	**1.29**	**2.76**	**2.6536**	1.56	3.52	3.3011	1.75	4.02	3.7757	1.92	4.40	4.1874
T-GCN + GRAFH (COM)	**1.29**	2.83	2.6537	**1.52**	**3.48**	**3.1729**	**1.71**	3.96	**3.6046**	1.89	4.37	**4.0108**
T-GCN + GRAFH (DEF)	**1.29**	2.83	2.7166	1.53	**3.48**	3.2488	**1.71**	**3.95**	3.6761	**1.87**	**4.33**	4.0663
STGNN	**1.20**	**2.60**	**2.2904**	1.53	3.53	**3.0559**	1.77	**4.13**	3.6311	1.97	4.58	4.1303
STGNN + GRAFH (COM)	1.21	2.63	2.4026	**1.52**	**3.52**	3.0981	**1.76**	**4.13**	3.6303	1.97	4.60	4.1202
STGNN + GRAFH (DEF)	1.24	2.65	2.3528	1.55	3.55	3.0765	1.77	**4.13**	**3.6225**	**1.96**	**4.57**	**4.1131**
Z-GCNETs	**1.19**	**2.77**	2.6004	1.42	**3.46**	3.1891	1.55	**3.85**	3.5333	**1.66**	**4.14**	3.8191
Z-GCNETs + GRAFH (COM)	**1.19**	2.78	**2.5441**	**1.41**	3.49	**3.1276**	**1.54**	3.88	**3.4780**	**1.65**	4.21	3.7879
Z-GCNETs + GRAFH (DEF)	1.22	2.86	2.6217	1.44	3.51	3.1892	1.56	3.90	3.5314	1.66	4.16	**3.7811**
AGCRN	1.18	2.66	2.4681	1.43	3.50	3.1518	1.58	3.99	3.5753	**1.71**	4.33	3.9113
AGCRN + GRAFH (COM)	**1.17**	2.62	2.4508	1.43	3.51	3.2027	1.58	4.02	3.6322	**1.71**	4.35	3.9511
AGCRN + GRAFH (DEF)	**1.17**	**2.61**	**2.3663**	**1.42**	**3.41**	**3.0809**	**1.57**	**3.89**	**3.5314**	**1.71**	**4.24**	**3.8838**

Declaration of Generative AI and AI-assisted technologies in the writing process

During the revision stage of this work, the author used ChatGPT in order to improve the language and readability of two previously written paragraphs. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the publication.

References

Baggag, A., Abbar, S., Sharma, A., Zanouda, T., Al-Homaid, A., Mohan, A., et al. (2021). Learning spatiotemporal latent factors of traffic via regularized tensor factorization: Imputing missing values and forecasting. *IEEE Transactions on Knowledge and Data Engineering*, *33*(6), 2573–2587.

Bai, L., Yao, L., Li, C., Wang, X., & Wang, C. (2021). Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in neural information processing systems* (pp. 17804–17815).

Chen, Y., Segovia, I., & Gel, Y. R. (2021). Z-GCNETs: Time zigzags at graph convolutional networks for time series forecasting. In *International conference on machine learning* (pp. 1684–1694).

Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing* (pp. 1724–1734).

Cohen, T., & Welling, M. (2016). Group equivariant convolutional networks. In *International conference on machine learning* (pp. 2990–2999).

Diao, Z., Wang, X., Zhang, D., Liu, Y., Xie, K., & He, S. (2019). Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 890–897).

Dong, S., Zhan, J., Hu, W., Mohajer, A., Bavaghar, M., & Mirzaei, A. (2023). Energy-efficient hierarchical resource allocation in uplink–downlink decoupled NOMA HetNets. *IEEE Transactions on Network and Service Management*, *20*(3), 3380–3395.

Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 922–929).

Guo, S., Lin, Y., Li, S., Chen, Z., & Wan, H. (2019). Deep spatial–temporal 3D convolutional neural networks for traffic data forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *20*(10), 3913–3926.

Guo, S., Lin, Y., Wan, H., Li, X., & Cong, G. (2022). Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*, *34*(11), 5415–5428.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780.

Huang, R., Huang, C., Liu, Y., Dai, G., & Kong, W. (2021). LSGCN: long short-term traffic prediction with graph convolutional networks. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 2355–2361).

Jia, R., Jiang, P., Liu, L., Cui, L., & Shi, Y. (2017). Data driven congestion trends prediction of urban transportation. *IEEE Internet of Things Journal*, *5*(2), 581–591.

Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, *207*, Article 117921.

Jiang, R., Wang, Z., Yong, J., Jeph, P., Chen, Q., Kobayashi, Y., et al. (2023). Spatio-temporal meta-graph learning for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence, vol. 37, no. 7* (pp. 8078–8086).

Jiang, J., Wu, B., Chen, L., Zhang, K., & Kim, S. (2023). Enhancing the robustness via adversarial learning and joint spatial-temporal embeddings in traffic forecasting. In *Proceedings of the 32nd ACM international conference on information and knowledge management* (pp. 987–996).

Jin, M., Zheng, Y., Li, Y., Chen, S., Yang, B., & Pan, S. (2022). Multivariate time series forecasting with dynamic graph neural ODEs. *IEEE Transactions on Knowledge and Data Engineering*, *35*, 9168–9180.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*.

Kong, Q.-J., Xu, Y., Lin, S., Wen, D., Zhu, F., & Liu, Y. (2013). UTN-model-based traffic flow prediction for parallel-transportation management systems. *IEEE Transactions on Intelligent Transportation Systems*, *14*(3), 1541–1547.

Lablack, M., & Shen, Y. (2023). Spatio-temporal graph mixformer for traffic forecasting. *Expert Systems with Applications*, *228*, Article 120281.

Lan, S., Ma, Y., Huang, W., Wang, W., Yang, H., & Li, P. (2022). Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *International conference on machine learning* (pp. 11906–11917). PMLR.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, F., Feng, J., Yan, H., Jin, G., Jin, D., & Li, Y. (2022). Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Transactions on Knowledge Discovery from Data*, *17*, 1–21.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International conference on learning representations*.

Liao, B., Zhang, J., Wu, C., McIlwraith, D., Chen, T., Yang, S., et al. (2018). Deep sequence learning with auxiliary information for traffic prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 537–546).

Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., & Zhao, W. (2017). A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, *4*(5), 1125–1142.

Lv, Z., Xu, J., Zheng, K., Yin, H., Zhao, P., & Zhou, X. (2018). LC-RNN: a deep learning model for traffic speed prediction. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 3470–3476).

Mohajer, A., Sam Daliri, M., Mirzaei, A., Ziaeddini, A., Nabipour, M., & Bavaghar, M. (2023). Heterogeneous computational resource allocation for NOMA: Toward green mobile edge-computing systems. *IEEE Transactions on Services Computing*, *16*(2), 1225–1238.

Mohajer, A., Sorouri, F., Mirzaei, A., Ziaeddini, A., Rad, K. J., & Bavaghar, M. (2022). Energy-aware hierarchical resource management and backhaul traffic optimization in heterogeneous cellular networks. *IEEE Systems Journal*, *16*(4), 5188–5199.

Nguyen, T. D., Nguyen, D. Q., Phung, D., et al. (2018). A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: Human language technologies, volume 2 (short papers)* (pp. 327–333).

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*(3), 1181–1191.

Shi, Z., Zhang, Y., Wang, J., Qin, J., Liu, X., Yin, H., et al. (2023). DAGCRN: Graph convolutional recurrent network for traffic forecasting with dynamic adjacency matrix. *Expert Systems with Applications*, *227*, Article 120259.

Snyder, C., & Do, M. N. (2019). STREETS: a novel camera network dataset for traffic flow. In *Proceedings of the 33rd international conference on neural information processing systems* (pp. 10242–10253).

Song, C., Lin, Y., Guo, S., & Wan, H. (2020). Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 914–921).

Sun, Y., Jiang, X., Hu, Y., Duan, F., Guo, K., Wang, B., et al. (2022). Dual dynamic spatial-temporal graph convolution network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, *23*(12), 23680–23693.

Tang, X., Yao, H., Sun, Y., Aggarwal, C., Mitra, P., & Wang, S. (2020). Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 5956–5963).

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Neural information processing systems* (pp. 5998–6008).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International conference on learning representations*.

Wang, J., Chen, R., & He, Z. (2019). Traffic speed prediction for urban transportation network: A path based deep learning approach. *Transportation Research Part C (Emerging Technologies)*, *100*, 372–385.

Wang, X., Ma, Y., Wang, Y., Jin, W., Wang, X., Tang, J., et al. (2020). Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of the world wide web conference* (pp. 1082–1092).

Welling, M., & Kipf, T. N. (2016). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., & Zhang, C. (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* (pp. 753–763).

Wu, Z., Pan, S., Long, G., Jiang, J., & Zhang, C. (2019). Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 1907–1913).

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., et al. (2018). Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI conference on artificial intelligence*.

Ye, Y., Xiao, Y., Zhou, Y., Li, S., Zang, Y., & Zhang, Y. (2023). Dynamic multi-graph neural network for traffic flow prediction incorporating traffic accidents. *Expert Systems with Applications*, *234*, Article 121101.

Yu, J. J. Q. (2022). Graph construction for traffic prediction: A data-driven approach. *IEEE Transactions on Intelligent Transportation Systems*, *23*(10), 15015–15027.

Yu, J. J. Q. (2023). Citywide estimation of travel time distributions with Bayesian deep graph learning. *IEEE Transactions on Knowledge and Data Engineering*, *35*, 2366–2378.

Yu, J. J. Q., Markos, C., & Zhang, S. (2021). Long-term urban traffic speed prediction with deep learning on graphs. *IEEE Transactions on Intelligent Transportation Systems*, *23*(7), 7359–7370.

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 3634–3640).

Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., & Chen, C. (2011). Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *12*(4), 1624–1639.

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., et al. (2019). T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, *21*(9), 3848–3858.

Zheng, C., Fan, X., Wang, C., & Qi, J. (2020). Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 1234–1241).

Zhu, L., Yu, F. R., Wang, Y., Ning, B., & Tang, T. (2018). Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *20*(1), 383–398.