FOGS: First-Order Gradient Supervision with Learning-based Graph for Traffic Flow Forecasting

Xuan Rao[1*], **Hao Wang**[2*], **Liang Zhang**[3], **Jing Li**[4], **Shuo Shang**[1†] and **Peng Han**[5†]
[1]University of Electronic Science and Technology of China, Chengdu, China
[2]Wuhan University, China
[3] King Abdullah University of Science and Technology, Saudi Arabia
[4] Inception Institute of Artificial Intelligence, Abu Dhabi, United Arab Emirates
[5]Aalborg University, Denmark
{raoxuanzzz, wanghao.hku, jedi.shang}@gmail.com, liangzhang@kaust.edu.sa,
jingli.phd@hotmail.com, pengh@cs.aau.dk

Abstract

Traffic flow forecasting plays a vital role in the transportation domain. Existing studies usually manually construct correlation graphs and design sophisticated models for learning spatial and temporal features to predict future traffic states. However, manually constructed correlation graphs cannot accurately extract the complex patterns hidden in the traffic data. In addition, it is challenging for the prediction model to fit traffic data due to its irregularly-shaped distribution. To solve the above-mentioned problems, in this paper, we propose a novel learning-based method to learn a spatial-temporal correlation graph, which could make good use of the traffic flow data. Moreover, we propose First-Order Gradient Supervision (FOGS), a novel method for traffic flow forecasting. FOGS utilizes first-order gradients, rather than specific flows, to train prediction model, which effectively avoids the problem of fitting irregularly-shaped distributions. Comprehensive numerical evaluations on four real-world datasets reveal that the proposed methods achieve state-of-the-art performance and significantly outperform the benchmarks.

1 Introduction

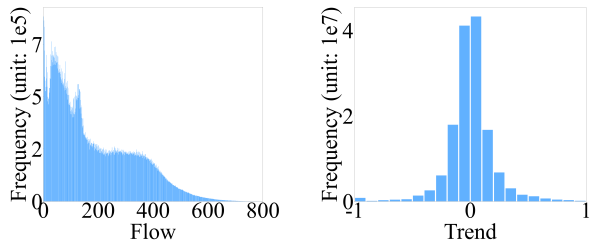
Traffic flow prediction is one of the most fundamental techniques for intelligent transportation management and services (e.g., route planning, intelligent traffic light control, etc.) [Wu and Tan, 2016; Zhang *et al.*, 2020; He and Shin, 2020]. Accurate prediction of future traffic conditions may help people make travel arrangements, avoid potential congestion in the streets, and wisely allocate transportation resources [Zhang *et al.*, 2017; Zhou *et al.*, 2021; Zhou *et al.*, 2018; Gong *et al.*, 2020; Liu *et al.*, 2021]. Typically, traffic flow prediction relies on traffic flow sensors, which are devices distributed in the road network monitoring the presence or passage of vehicles. A sensor records traffic flows at a fixed frequency (e.g., every 5 minutes), thus generating a sequence of traffic flows over time. Essentially, the data adopted for prediction is a sequence of traffic flow signals, where each signal contains the traffic flows recorded by all the sensors in the road network during some time intervals. The task of traffic flow prediction is naturally *spatiotemporal*: to predict the future flows of some sensors [Lv *et al.*, 2015; Yu *et al.*, 2017; Bai *et al.*, 2019; Yao *et al.*, 2018a; Yao *et al.*, 2018b] , one may need to consider the flows in the past as well as spatial correlations such as geographical distances to nearby sensors, and structure of the road network.

Many approaches [Fang *et al.*, 2019; Li *et al.*, 2021; Ye *et al.*, 2021; Wu *et al.*, 2019; Fang *et al.*, 2021] have been proposed for traffic flow prediction. In a nutshell, these methods first build a *correlation graph* between sensors and then make traffic flow predictions based on the correlation graph. STGCN [Yu *et al.*, 2018] directly uses the road network to describe the spatial correlations between sensors and uses a graph convolution method to extract spatiotemporal features. STSGCN [Song *et al.*, 2020] explicitly establishes temporal correlations between sensors in consecutive traffic flow signals and then uses graph embedding techniques for flow prediction. Recently, [Li and Zhu, 2021] models correlations between sensors via similarities between their flow time series. They construct a fusion graph to reflect both temporal and spatial correlations and propose a gated dilated CNN module for flow prediction.

Although existing methods have made achievements in their own lines of research, we observe the following issues:

- *Graph building*. Most existing methods fail to make full use of the temporal information in the traffic data. They either completely ignore the historical flows or merely use some overall temporal similarity between them. There lacks consideration on temporal similarity of historical flows at finer granularities. Moreover, most existing correlation graphs are manually built, thus human experience may largely bias the prediction.
- *Prediction making*. Most existing methods aim to predict the exact traffic flows. The task is challenging as the distribution of traffic flows is irregularly shaped. Us-

[*]Indicates equal contribution
[†]Corresponding Author



(a) Flow distribution (b) Trend distribution

Figure 1: The comparison between the flow distribution and the trend distribution of the training set of the PEMS03 dataset.

ing a limited amount of training data usually causes the problem of under-fitting, which may degrade the accuracy of flow prediction.

In this paper, we aim to tackle the above issues. To make better use of the temporal information, we observe that the traffic flow usually follows a regular weekly pattern, reflecting human activities in weekdays and weekends, rush hours and regular hours, etc. Based on this observation, we decompose a week into a sequence of consecutive time slots, each corresponding to a specific time interval (e.g., 0:00 to 0:05 A.M. on Wednesday) in a week. For each sensor, we construct a *temporal feature vector*, where each element is the average of the historical flows within the corresponding time slot. The distance between two temporal feature vectors is thereby a similarity measure between the corresponding sensors. With such a similarity measure, we build a *temporal correlation graph* by linking each sensor with its k most similar sensors. Then, we propose a novel learning-based approach to learn an *embedding* of each sensor, considering both the temporal correlation graph and the road network, which can be used for accurate flow prediction.

To tackle the underfitting problem during prediction making, we design a novel method named **First-Order Gradient Supervision (FOGS)**. FOGS employs the first-order gradients, a.k.a. *trends*, rather than the exact flows to train the prediction model. Briefly speaking, at some moment, the trend is the relative temporal change of the flow from the previous moment. We show the distributions of flows and trends of the training set of the PEMS03 dataset[1] in Figure 1. As we can see, flows and trends have very different distributional properties. The flow distribution is irregularly shaped and widely spread, while the trend distribution is well concentrated. This is consistent with our intuition that the human activities follow some regular patterns in general, although the exact flows may differ from one day to another. Therefore, we argue that trends will provide more supervision for flow prediction. FOGS aims to make wise use of such supervision from trends.

In summary, we make the following contributions to traffic flow prediction:

- We propose a novel graph embedding approach to learn a data representation for each sensor. Comparing to existing solutions, our data representation better reflects both temporal and spatial correlations between sensors.
- We reveal the importance of trends in flow prediction. We propose a novel method, FOGS, which can make wise use of the supervision information provided by trends for accurate flow prediction.
- We conduct extensive experiments on four real-world datasets to test the performance of FOGS. The results show that FOGS significantly outperforms the existing solutions in terms of accuracy.

2 Related Work

Recently, graph convolution methods have been widely used to model the spatiotemporal correlations in traffic network data. DCRNN [Li *et al.*, 2018] employs bi-directional random walks to characterize the diffusion process regarding spatial relationships. It combines gated recurrent units (GRU) with diffusion convolution, and then proposes an encoder-decoder model for temporal correlations. STGCN [Yu *et al.*, 2018] utilizes convolutional structures on spatial and temporal domain to extract spatiotemporal features simultaneously. ASTGCN [Guo *et al.*, 2019] proposes to utilize spatial and temporal attention mechanism to learn spatial and temporal correlations, respectively. STSGCN [Song *et al.*, 2020] employs spatiotemporal synchronous graph convolutional module to directly capture the localized spatiotemporal correlations. STFGNN [Li and Zhu, 2021] uses the dynamic time warping (DTW) algorithm to construct a temporal graph and merges it with a given spatial graph into a novel spatiotemporal fusion graph. As mentioned in Section 1, the graphs used by these methods are manually built, thus human experience may affect the prediction performance.

3 Preliminaries and Problem Statement

In this section, we formally introduce the traffic flow prediction problem with necessary definitions and preliminaries.

3.1 Definitions

Road network. A road network is an undirected graph $\mathcal{R} = (V, E_{\text{road}})$, where V and E_{road} are the sets of sensors and the road segments between sensors, respectively. The road network $\mathcal{R}$ indicates the spatial relationship between sensors, which remains static over time.

Traffic flow & graph signal. Given a road network $\mathcal{R} = (V, E_{\text{road}})$ and a sequence of T consecutive time intervals, the observed *graph signal* on $\mathcal{R}$ during the t-th time interval is a nonnegative vector $\boldsymbol{x}_{\mathcal{R}}^{(t)} \in \mathbb{R}^{|V|}$, where the j-th element of $\boldsymbol{x}_{\mathcal{R}}^{(t)}$ is the *traffic flow* observed by the j-th sensor during the t-th time interval.

3.2 Problem Statement

At time t, given a road network $\mathcal{R}$ and previous T graph signals $\boldsymbol{x}_{\mathcal{R}}^{(t-T+1)}, \cdots, \boldsymbol{x}_{\mathcal{R}}^{(t)}$, the problem of *traffic flow predic-*

[1]This dataset contains traffic flow data recorded by 358 sensors from 1 Sept 2018 to 30 Nov 2018 in California. See Section 6 for further details.

tion is to find a function f to predict future K graph signals:

$$\left[\boldsymbol{x}_{\mathcal{R}}^{(t-T+1)}, \cdots, \boldsymbol{x}_{\mathcal{R}}^{(t)}\right] \xrightarrow{f} \left[\boldsymbol{x}_{\mathcal{R}}^{(t+1)}, \cdots, \boldsymbol{x}_{\mathcal{R}}^{(t+K)}\right].$$

4 Spatiotemporal Correlation Learning

In this section, we present our method for learning the spatiotemporal correlations between sensors. The method works with a road network $\mathcal{R}$, which reflects spatial correlations, and a temporal correlation graph $\mathcal{C}$ to generate a spatiotemporal representation for each sensor. By properly defining a neighborhood based on the learned representation, we obtain a graph depicting the spatiotemporal correlations between sensors, which can then be used for flow prediction.

4.1 Temporal Correlation Graph

Formally, the temporal correlation graph is $\mathcal{C} = (V, E_{\text{time}})$, where V is the set of sensors, as in the road network $\mathcal{R}$, and E_{time} represents temporal correlations between sensors. As explained in Section 1, when establishing E_{time}, we wish to make use of the temporal patterns at finer granularities.

To that end, we decompose a week into a sequence of equally-lengthed consecutive time slots, each corresponding to a certain time interval in a week (e.g., 0:00 to 0:05 A.M. on Wednesday). The length of each time slot is $\frac{1}{\omega}$, where ω is the data collection frequency of sensors. Suppose that a week consists of N_ω such time slots. Then, for each sensor v, we construct an N_ω-dimensional feature vector, where the j-th element is the average of all the flows ever recorded by sensor v during the j-th time slot in a week.

Using such temporal feature vectors, $(u, v) \in E_{\text{time}}$ if v is a k-nearest neighbor of u, where the neighborhood is decided by measures such as the Euclidean distance.

4.2 Objective Function

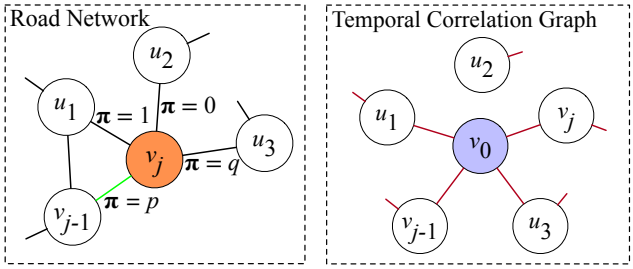
Given the road network $\mathcal{R} = (V, E_{\text{road}})$ and the temporal correlation graph $\mathcal{C} = (V, E_{\text{time}})$, our next goal is to find an *embedding function* $\boldsymbol{h} : V \to \mathbb{R}^m$ that maps each sensor $v \in V$ into an m-dimensional feature vector (i.e., embedding). The embedding function $\boldsymbol{h}(\cdot)$ is expected to preserve the closeness between sensors in both $\mathcal{R}$ and $\mathcal{C}$. In this sense, we essentially want to simultaneously learn an embedding function $\boldsymbol{h}$ and a correlation graph $\mathcal{G} = (V, E)$, where the spatiotemporal correlations between sensors are well reflected in E and well preserved by $\boldsymbol{h}$.

To implement the learning, we use the skip-gram method [Mikolov *et al.*, 2013] to capture the "neighborhood" properties. The learning objective used in our work is

$$\max_{\boldsymbol{h}} \sum_{v \in V} \log \Pr\left(\mathcal{N}_S(v) | \boldsymbol{h}(v)\right). \tag{1}$$

Here, $\mathcal{N}_S(v) \subseteq V$ is the "neighbors" of sensor v, which, in our context, refers to all the sensors reachable from v under some *sampling strategy* S. $\Pr\left(\mathcal{N}_S(v) | \boldsymbol{h}(v)\right)$ is the probability of observing $\mathcal{N}_S(v)$ given the embedding $\boldsymbol{h}(v)$. Assuming probabilistic independence, this probability can be calculated as

$$\Pr\left(\mathcal{N}_S(v) | \boldsymbol{h}(v)\right) = \prod_{u \in \mathcal{N}_S(v)} \Pr\left(u | \boldsymbol{h}(v)\right),$$



(a) Random walk (b) Traffic flow correlation

Figure 2: A simple illustration of our random walk sampling strategy. (a) The walk just traversed the edge (v_{j-1}, v_j) and stops at sensor v_j now in the road network $\mathcal{R}$. Then we want to choose the next sensor v_{j+1}. (b) v_0 is the starting sensor of the walk, and the traffic flow pattern of v_0 is similar to that of v_{j-1}, v_j, u_1, u_3 but not to u_2 in the temporal correlation graph $\mathcal{C}$.

where a common choice for $\Pr\left(u | \boldsymbol{h}(v)\right)$ is

$$\Pr\left(u | \boldsymbol{h}(v)\right) = \frac{\exp\left(\boldsymbol{h}(u)^\top \boldsymbol{h}(v)\right)}{\sum_{w \in V} \exp\left(\boldsymbol{h}(w)^\top \boldsymbol{h}(v)\right)}.$$

Consequently, the objective becomes

$$\max_{\boldsymbol{h}} \sum_{v \in V} \left(-\log \Psi(v) + \sum_{u \in \mathcal{N}_S(v)} \boldsymbol{h}(u)^\top \boldsymbol{h}(v) \right), \tag{2}$$

where $\Psi(v) = \sum_{w \in V} \exp\left(\boldsymbol{h}(w)^\top \boldsymbol{h}(v)\right)$.

4.3 Optimizing the Objective Function

Figure 2 illustrates our random walk process. Clearly, the objective function of Eq. 2 is largely affected by the neighborhood function $\mathcal{N}_S(\cdot)$, which is in turn determined by the sampling strategy S. Direct computation of the objective function may be prohibitive in practice, as we need to traverse V, the entire set of sensors, to compute the sum $\Psi(v)$ during the training process. Such a traversal should be done every time the embedding function $\boldsymbol{h}(\cdot)$ is updated, since there is no easy way to reuse the results.

The problem can be solved using *random walks* and *negative sampling* [Grover and Leskovec, 2016; Han *et al.*, 2021]. Starting from a source sensor $v_0 \in V$, a random walk of length L generates a path $\tau = \langle v_0, v_1, \cdots, v_L \rangle$. With a predefined threshold Δ ($\Delta \ll L$), we may obtain a set of neighbors of v_j from the path τ,

$$\mathcal{N}_\tau(v_j) = \{ v_i \in \tau \,||i - j| \le \Delta, i \ne j \},$$

where $j = 0, 1, \cdots, L$. Essentially, $\mathcal{N}_\tau(v_j)$ is a sample of $\mathcal{N}_S(v_j)$, the sensors in which are intuitively closer to v_j than the others are. Such discrimination can then be fed into optimization algorithms such as stochastic gradient descent (SGD) to find the optimal embedding $\boldsymbol{h}$.

Conventional graph embedding methods generate random walks based on the topological structure to find the set of neighbors, which makes no use of the temporal correlations. In this work, we propose a new sampling strategy S to take temporal correlations into account in random walks.

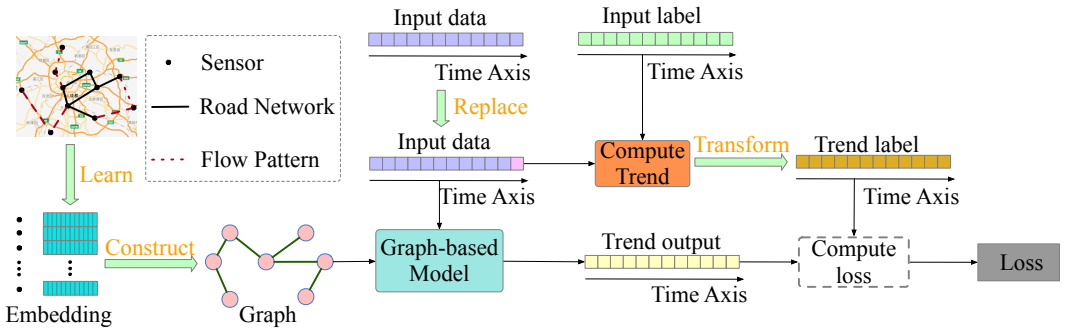


Figure 3: The framework of FOGS. We first learn the embeddings of sensors and then use the embeddings to construct a learning-based graph $\mathcal{G}$. Input data $\boldsymbol{X}$ and graph $\mathcal{G}$ are fed into graph-based model g to get the predicted trend $\hat{\boldsymbol{Z}}$, which is used together with the real trend matrix $\boldsymbol{Z}$ to calculate the loss.

Let $\mathcal{R} = (V, E_{\text{road}})$ and $\mathcal{C} = (V, E_{\text{time}})$ be the road network and the temporal correlation graph, respectively. Assume that a random walk starts from sensor v_0 has so far generated a path $\tau_j = \langle v_0, \cdots, v_{j-1}, v_j \rangle$. (Thus, the random walk is now at sensor v_j.) The sampling strategy S decides the next sensor v_{j+1} to visit based on the probability:

$$\Pr(v_{j+1}|\tau_j) \propto \begin{cases} \pi(\tau_j, v_{j+1}), & \text{if } (v_j, v_{j+1}) \in \mathcal{R}, \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\pi(\tau_j, v_{j+1})$ is a weight assigned to the (potential) edge (v_j, v_{j+1}):

$$\pi(\tau_j, v_{j+1}) = \begin{cases} p, & \text{if } d_{\mathcal{R}}(v_{j+1}) = 0 \text{ and } (v_0, v_{j+1}) \in \mathcal{C}, \\ 1, & \text{if } d_{\mathcal{R}}(v_{j+1}) = 1 \text{ and } (v_0, v_{j+1}) \in \mathcal{C}, \\ q, & \text{if } d_{\mathcal{R}}(v_{j+1}) = 2 \text{ and } (v_0, v_{j+1}) \in \mathcal{C}, \\ 0, & \text{otherwise,} \end{cases}$$

where $d_{\mathcal{R}}(v_{j+1})$ is the shortest path distance between v_{j+1} and $v_{j-1} \in \tau_j$ in the road network $\mathcal{R}$.

There are three choices for the next sensor v_{j+1}: (i) sensor v_{j-1}, (ii) sensor u_1 near v_{j-1}, and (iii) sensor u_2 or u_3 away from v_{j-1}. Therefore, $d_{\mathcal{R}}(v_{j+1})$ must be 0, 1, or 2. In this way, we could guarantee that the traffic flow patterns of all the sensors in a random walk is similar to that of the source sensor v_0. Meanwhile, the proposed sampling method can retain the topology structure of the road network as well.

5 First-Order Gradient Supervision

In this section, we present FOGS, our novel traffic flow supervision method.

5.1 First-Order Gradient

Existing studies typically utilize the exact traffic flows to train their models and then predict the future flows in the network. However, we argue that there are many external factors affecting the traffic flow at a particular sensor, such as geographical location, facilities in the surrounding area, etc. As shown in Figure 1(a), the distribution of traffic flows is irregularly shaped, which is difficult to fit with limited amount of training data. Nonetheless, from Figure 1(b), we observe that the distribution of *trends* is concentrated.

Based on such observation, we propose a novel approach, namely FOGS, which focuses on trends instead of traffic flows. Recall that traffic flow prediction is to find a function f to predict future K graph signals based on the previous T ones. Therefore, from any data available, we have training samples of the form $\boldsymbol{X} \to \boldsymbol{Y}$, where

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T] \in \mathbb{R}^{|V| \times T}$$

is the matrix of T consecutive graph signals and

$$\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_K] \in \mathbb{R}^{|V| \times K}$$

is the matrix of the next K ones right after $\boldsymbol{x}_T$, i.e., $\boldsymbol{y}_t = \boldsymbol{x}_{T+t}$ for $t = 1, 2, \cdots, K$.

For the j-th sensor $v_j \in V$, we define the *trend* of the traffic flow at v_j at time $T + t$, denoted z_{jt}, as follows:

$$z_{jt} = \frac{y_{jt} - x_{jT}}{x_{jT}}. \quad (t = 1, 2, \cdots, K) \tag{3}$$

The trend z_{jt} is the relative change of y_{jt} over x_{jT}, the last observed flow by sensor v_j. In occasional cases, $x_{jT} = 0$. Then, we look backwards in time, scanning $x_{j(T-1)}, x_{j(T-2)}, \cdots, x_{j1}$ in order for the first nonzero value and use it as the reference for the relative change. [2] Eventually, given a training sample $\boldsymbol{X} \to \boldsymbol{Y}$, we obtain a trend matrix $\boldsymbol{Z}$ of the same shape of $\boldsymbol{Y}$.

5.2 Spatiotemporal Correlation Graph

Since we have learned an embedding function $\boldsymbol{h}(\cdot)$ over the sensors V, which well preserves both the spatial and temporal correlations, we are now ready to construct a learning-based spatiotemporal correlation graph $\mathcal{G}$.

To do this, we first compute a spatiotemporal correlation matrix $\boldsymbol{M} \in [0, 1]^{|V| \times |V|}$ between each pair of sensors. Specifically, the (i, j)-th entry of $\boldsymbol{M}$ is defined as

$$\boldsymbol{M}(i, j) = \cos(\boldsymbol{h}(v_i), \boldsymbol{h}(v_j)),$$

where $i, j = 1, 2, \cdots, |V|$.

[2] There is still chance that $x_{jt} \equiv 0$ for every $t = 1, 2, \cdots, T$. In this case, we replace x_{jt} with first nonzero flow average recorded by sensor j in same way. However, the flow averages could still be zero, we choose to replace x_{jt} with constant $\sigma = 5$.

Based on $\boldsymbol{M}$, we may calculate a k-nearest neighbor set for each sensor v_j, using, for example, the Euclidean distance. Then, we have

$$\boldsymbol{G}(i,j) = \begin{cases} \boldsymbol{M}(i,j), & \text{if } v_j \text{ is a } k \text{ nearest neighbor of } v_i, \\ 0, & \text{otherwise.} \end{cases}$$

Finally, by normalizing the matrix $\boldsymbol{G}$, we obtain the learned spatiotemporal correlation graph $\mathcal{G}$:

$$\mathcal{G} = \boldsymbol{D}^{-1}\boldsymbol{G},$$

where $\boldsymbol{D} = \mathbf{diag}\,(\boldsymbol{G})$ denotes the out-degree diagonal matrix derived from $\boldsymbol{G}$.

5.3 Model Framework

Figure 3 shows the FOGS framework. The replaced input data $\boldsymbol{X}$ and the spatiotemporal correlation graph $\mathcal{G}$ are fed into the graph-based model g to obtain the prediction output, i.e., the trend matrix:

$$g(\boldsymbol{X}, \mathcal{G}) = \hat{\boldsymbol{Z}}. \tag{4}$$

where g denotes a prediction model such as STFGNN.

We choose MAE loss as the loss function, which is defined as follows:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{Z}, \hat{\boldsymbol{Z}}) = \frac{\sum_{i=1}^{|V|}\sum_{j=1}^{K} |\boldsymbol{Z}_{ij} - \hat{\boldsymbol{Z}}_{ij}|}{|V| \times K}. \tag{5}$$

During evaluation phase, we convert the predicted trends $\hat{\boldsymbol{Z}} \in \mathbb{R}^{|V|\times K}$ back into corresponding traffic flows $\hat{\boldsymbol{Y}}$:

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}_T \odot \hat{\boldsymbol{Z}} + \boldsymbol{X}_T, \tag{6}$$

where $\boldsymbol{X}_T$ is the $|V| \times K$ matrix with the traffic signal $\boldsymbol{x}_T$ duplicated K times, and $\odot$ is the Hadamard product.

6 Experiments

In this section, extensive experiments are conducted to show the effectiveness of our approach. We first introduce the datasets and the baseline methods. Then, we introduce the experimental results and analyses by comparing with the baselines. Finally, we conduct ablation experiment to show the effect of some components in our framework.

6.1 Datasets

We conduct experiments on four real-world datasets from [Song *et al.*, 2020]: PEMS03, PEMS04, PEMS07, and PEMS08. The data are extracted from Caltrans Performance Measurement System (PeMS) [Chen *et al.*, 2001] in four different districts in California. The raw data are aggregated into 5-minute time intervals, thus there are 12 points in the traffic data per hour. Each sensor in PEMS04 and PEMS08 has three traffic-related data: flow, occupancy, and speed. In this paper, we use the flow data and ignore the rest. Table 1 shows the statistics of the four datasets used in our experiments.

Same as what previous studies [Song *et al.*, 2020; Li and Zhu, 2021] do, we map each sensor v into real road networks to construct the the road network of sensors, i.e., $\mathcal{R}$ in Section 3. In addition, we utilize Z-score normalization to standardize the input data.

Dataset	#Days	#Sensors	#Edges	#Data
PEMS03	91	358	547	26208
PEMS04	59	307	340	16992
PEMS07	98	883	866	28224
PEMS08	62	170	295	17856

Table 1: Datasets

6.2 Settings

We split these four datasets with ratio $7:1:2$ into training, validation, and test sets. We use the flows of one hour to predict the ones in the next hour. That is, we set $T = K = 12$ in our experiments, using 12 continuous graph signals to predict the next 12 ones. The parameters p and q in our random walk strategy, as introduced in Section 4, are both set to 1. When constructing the temporal correlation graph $\mathcal{C}$ (Section 4) and the final spatiotemporal correlation graph $\mathcal{G}$ (Section 5), we consider $k = 10$ nearest neighbors for each sensor. Moreover, we set the dimension of sensor embeddings to 128 , the random walk length L to 25, and the window threshold Δ to 10. Our implementation is available in Pytorch [3].

6.3 Baselines

We consider the following five state-of-the-art methods as the baselines in the experiments.

- ASTGCN [Guo *et al.*, 2019]: Attention Based Spatial Temporal Graph Convolutional Networks, which models spatial and temporal correlations with spatial and temporal attention mechanisms. To conduct a fair comparison, only the recent components are used to model the periodicity of the traffic data.
- DCRNN [Li *et al.*, 2018]: Diffusion Convolutional Recurrent Neural Network, which incorporates encoder-decoder model with diffusion graph convolution.
- STGCN [Yu *et al.*, 2018]: Spatial-Temporal Graph Convolutional Networks, which only utilizes convolutional structures on spatial and temporal domains to extract spatiotemporal features simultaneously.
- STSGCN [Song *et al.*, 2020]: Spatial-Temporal Synchronous Graph Convolutional Networks, which constructs a localized spatiotemporal graph to capture spatiotemporal correlations simultaneously.
- STFGNN [Li and Zhu, 2021]: Spatial-Temporal Fusion Graph Neural Networks, which designs a spatiotemporal fusion graph. A fusion graph neural module and a gated convolution module are combined to model spatiotemporal correlations.

For our method FOGS, we choose STFGNN as the base prediction model.

6.4 Results and Analysis

The experiment results on the four public datasets are shown in Table 2. Based on these results, we have the following observations and corresponding analysis:

[3] https://github.com/kevin-xuan/FOGS

Datasets	Metric	ASTGCN	DCRNN	STGCN	STSGCN	STFGNN	FOGS **(ours)**
PEMS03	MAE	18.05	17.86	17.52	17.17	16.80	**15.06**
	MAPE(%)	17.02	18.30	17.08	16.17	16.23	**14.11**
	RMSE	30.13	29.74	30.23	28.58	28.44	**24.25**
PEMS04	MAE	21.85	23.54	22.72	20.98	19.85	**19.35**
	MAPE(%)	14.11	17.18	14.56	13.73	12.99	**12.71**
	RMSE	34.54	36.25	35.56	33.58	31.89	**31.33**
PEMS07	MAE	25.22	23.87	24.58	23.39	21.51	**20.62**
	MAPE(%)	11.41	10.50	10.65	9.83	9.02	**8.58**
	RMSE	38.83	37.27	37.51	37.79	35.67	**33.96**
PEMS08	MAE	18.70	18.41	18.14	16.35	16.75	**14.92**
	MAPE(%)	11.64	12.17	11.38	10.54	10.58	**9.42**
	RMSE	28.66	28.28	27.97	25.64	26.35	**24.09**

Table 2: Performance Comparison Against Baselines

- Our proposed methods significantly outperform all other state-of-the-art baselines under all metrics on all datasets. Specifically, FOGS outperforms ASTGCN, DCRNN, STGCN, STSGCN and STFGNN with ratios 19.52%, 18.46%, 19.78%, 15.15%, and 14.73%, respectively under the RMSE metric on PEMS03 dataset. This shows the superiority of our framework to some extent.
- Comparing STFGNN and our method FOGS, we can find that our proposed methods greatly improve the performance. The reason might be that the spatiotemporal correlation graph $\mathcal{G}$ used in our method is learned, instead of manually constructed, which could better capture the spatiotemporal information. Moreover, it also demonstrates that training models with trends instead of traffic flows could enhance the models performance.
- Comparing the performance improvement on the PEMS04 ans PEMS07 datasets, we could see that FOGS performs better on the PEMS03 and PEMS08 datasets. The reason might be that the road network is more sparse on PEMS04 and PEMS07 datasets, which leads to the inaccuracy of the learned graph $\mathcal{G}$. Moreover, a higher rate of missing traffic data will also result in poor model performance.

6.5 Ablation Experiment

There are two main components in our framework: (i) the learned graph, and (ii) the trend supervision. To show the effects of these components, we conduct ablation experiments on FOGS, of which the results are shown in Table 3.

From Table 3, we may have the following conclusions and analyses:

- We can find that our learned graph could improve the model performance. Because the sensor embedding learned through the random walk algorithm can not only reflect the topology of the spatial network, but also reflect historic traffic flow patterns. The combination of them allows the learned graph to make effective use of the side information.
- We could notice that training STFGNN model with trend could improve model performance on all datasets. Because trends tend to be consistent and concentrated, training model with trends can significantly enhance the model performance.

Datasets	Metric	Baseline	Graph	Trend
PEMS03	MAE	16.80	15.53	**15.06**
	MAPE(%)	16.23	14.82	**14.11**
	RMSE	28.44	27.24	**24.25**
PEMS04	MAE	19.85	19.48	**19.34**
	MAPE(%)	12.99	12.81	**12.71**
	RMSE	31.89	31.56	**31.20**
PEMS07	MAE	21.51	20.88	**20.62**
	MAPE(%)	9.02	8.80	**8.58**
	RMSE	35.67	34.40	**33.96**
PEMS08	MAE	16.75	15.47	**14.92**
	MAPE(%)	10.58	9.95	**9.42**
	RMSE	26.35	24.78	**24.09**

Table 3: FOGS Ablation Experiments. Baseline denotes the STFGNN method, Graph stands for training model with our learned graph, and Trend means FOGS method.

7 Conclusion

In this paper, we propose a novel learning-based approach to learn a graph that could make effective use of the information in spatio-temporal data. Embedding of every sensor is learned by our sampling method, which could capture the topology structure of the road network and historic traffic flow patterns. The graph is constructed by the cosine values between embeddings of sensors with k-nearest neighbors. Moreover, a novel supervised method, FOGS is presented to enhance model performance, which uses trend rather than specific flow to train models. We conduct extensive experiments and analysis on four public datasets, which show that our proposed approaches significantly outperform the existing baselines.

Acknowledgements

This work was supported by the NSFC (U2001212, 62032001, and 61932004).

References

[Bai *et al.*, 2019] Lei Bai, Lina Yao, Salil S. Kanhere, Xianzhi Wang, and Quan Z. Sheng. Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. In *IJCAI*, 2019.

[Chen *et al.*, 2001] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: Mining loop detector data. *TRR*, 2001.

[Fang *et al.*, 2019] Shen Fang, Qi Zhang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Gstnet: Global spatial-temporal network for traffic flow prediction. In *IJCAI*, 2019.

[Fang *et al.*, 2021] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ODE networks for traffic flow forecasting. In *KDD*, 2021.

[Gong *et al.*, 2020] Yongshun Gong, Zhibin Li, Jian Zhang, Wei Liu, and Jinfeng Yi. Potential passenger flow prediction: A novel study for urban transportation development. In *AAAI*, 2020.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *ACM SIGKDD*, 2016.

[Guo *et al.*, 2019] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, 2019.

[Han *et al.*, 2021] Peng Han, Jin Wang, Di Yao, Shuo Shang, and Xiangliang Zhang. A graph-based approach for trajectory similarity computation in spatial networks. In *ACM SIGKDD*, 2021.

[He and Shin, 2020] Suining He and Kang G. Shin. Towards fine-grained flow forecasting: A graph attention approach for bike sharing systems. In *WWW*, 2020.

[Li and Zhu, 2021] Mengzhang Li and Zhanxing Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *AAAI*, 2021.

[Li *et al.*, 2018] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

[Li *et al.*, 2021] Mingqian Li, Panrong Tong, Mo Li, Zhongming Jin, Jianqiang Huang, and Xian-Sheng Hua. Traffic flow prediction with vehicle trajectories. In *AAAI*, 2021.

[Liu *et al.*, 2021] Hao Liu, Qiyu Wu, Fuzhen Zhuang, Xinjiang Lu, Dejing Dou, and Hui Xiong. Community-aware multi-task transportation demand prediction. In *AAAI*, 2021.

[Lv *et al.*, 2015] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *TITS*, 2015.

[Mikolov *et al.*, 2013] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[Song *et al.*, 2020] 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 *AAAI*, 2020.

[Wu and Tan, 2016] Yuankai Wu and Huachun Tan. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *CoRR*, 2016.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, 2019.

[Yao *et al.*, 2018a] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, Yanwei Yu, and Zhenhui Li. Modeling spatial-temporal dynamics for traffic prediction. *CoRR*, 2018.

[Yao *et al.*, 2018b] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *AAAI*, 2018.

[Ye *et al.*, 2021] Junchen Ye, Leilei Sun, Bowen Du, Yanjie Fu, and Hui Xiong. Coupled layer-wise graph convolution for transportation demand prediction. In *AAAI*, 2021.

[Yu *et al.*, 2017] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. Deep learning: A generic approach for extreme condition traffic forecasting. In *ICDM*, 2017.

[Yu *et al.*, 2018] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, 2018.

[Zhang *et al.*, 2017] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, 2017.

[Zhang *et al.*, 2020] Junbo Zhang, Yu Zheng, Junkai Sun, and Dekang Qi. Flow prediction in spatio-temporal networks based on multitask deep learning. *TKDE*, 2020.

[Zhou *et al.*, 2018] Xian Zhou, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. Predicting multi-step citywide passenger demands using attention-based neural networks. In *WSDM*, 2018.

[Zhou *et al.*, 2021] Qiang Zhou, Jingjing Gu, Xinjiang Lu, Fuzhen Zhuang, Yanchao Zhao, Qiuhong Wang, and Xiao Zhang. Modeling heterogeneous relations across multiple modes for potential crowd flow prediction. In *AAAI*, 2021.