Dynamic Prediction of Origin-Destination Flows Using Fusion Line Graph Convolutional Networks

Xi Xiong, Kaan Ozbay, Li Jin, and Chen Feng *

May 2, 2019

Abstract

Modern intelligent transportation systems provide data that allow real-time demand prediction, which is essential for planning and operations. The main challenge of prediction of Origin-Destination (O-D) flow matrices is that demands cannot be directly measured by traffic sensors; instead, they have to be inferred from aggregate traffic flow data on traffic links. Specifically, spatial correlation, congestion and time dependent factors need to be considered in general transportation networks. In this paper we propose a novel O-D prediction framework based on Fusion Line Graph Convolutional Networks (FL-GCNs). We use FL-GCN to recognize spatial and temporal patterns simultaneously. The underlying road network topology is transformed into a corresponding line graph. This structure provides a general framework for predicting spatial-temporal O-D information from link traffic flows. Data from a New Jersey Turnpike network is used to evaluate the proposed model. The results show that FL-GCN can recognize spatial and temporal patterns. We also compare FL-GCN with Kalman filter; the results show that our model can outperform Kalman filter by 17.87% in predicting the whole O-D pairs.

Keywords: Graph Neural Network, demand prediction, machine learning.

1 Introduction

Origin-Destination (O-D) flows are fundamental prerequisites for transportation analysis. Reliable prediction of O-D flows can improve planning and operations in real-time traffic management [1, 2, 3]. Traffic demand is typically characterized by an O-D matrix. The elements in the matrix denote the number of trips between O-D pairs during a certain time interval. In some cases, link flows can be obtained from traffic sensors such as loop inductors (Fig. 1). Then we can use real-time link flows, historical O-D flows, historical link flows to forecast future O-D flows. Network topology is used in the process of parameter estimation and demand prediction.

Prediction of O-D flows has been studied for decades. Several models have been proposed to solve the problem. Gravity Model is a widely used approach to tackle static O-D prediction problems. However, its effectiveness is limited because of highly dynamic and nonlinear features of transportation flows that cannot be captured by its underlying mathematical structure. Statistical models, such as Generalized Least Squares (GLS), Maximum Likelihood (ML) estimation,

---

*X. Xiong, Kaan Ozbay, and Li Jin are with the Department of Civil and Urban Engineering and C2SMART University Transportation Center, and Chen Feng is joint appointment to Department of Civil and Urban Engineering and Department of Mechanical and Aerospace Engineering, New York University Tandon School of Engineering, Brooklyn, NY, USA, emails: xi.xiong@nyu.edu, kaan.ozbay@nyu.edu, lijin@nyu.edu, cfeng@nyu.edu.
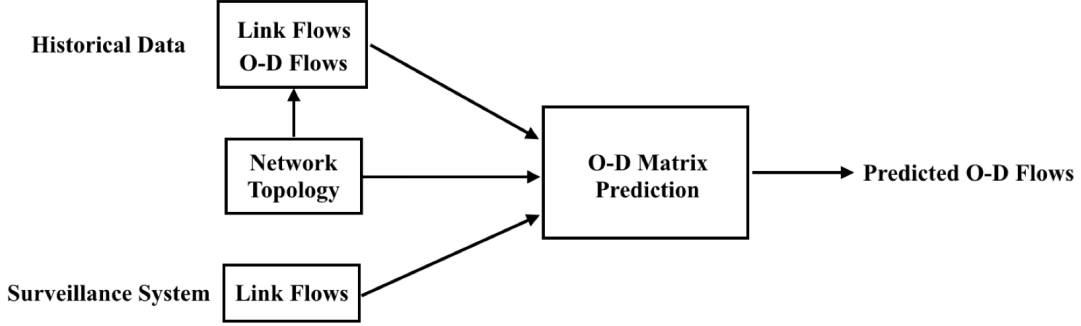
Fig. 1. Illustration of O-D flow prediction.

and Bayesian methods are widely used to solve the O-D estimation and prediction problem. The objective of GLS is to minimize the difference between estimated flows and observed flows [4, 5]. ML estimators are obtained by maximizing the likelihood of observed flows conditional on estimated O-D flows [6]. In the framework of Bayesian approach, posterior probability is calculated by combining prior probability expressed by O-D flows and link flow likelihood conditional on estimated O-D flows. The Bayesian solution [7, 8] is to find O-D flows that would maximize posterior probability.

Advanced models take spatial and temporal effects into consideration. For the spatial part, the key problem is the mapping from O-D flows to link flows. Assignment matrices are often used to represent this relationship. There are two steps from O-D flows to link flows. The first step is the mapping from O-D flows to path flows. We need to consider route choice behaviors in this step. The second step is from path flows to link flows. If the path is not congested, the mapping of path to link flows is given by a link-path incidence matrix. In general transportation networks with congestion, User Equilibrium (UE) is often used to characterize route choice behaviors. The bi-level O-D estimation method incorporates the UE assumption [9]. The upper level is to minimize the difference between estimated and observed flows. The lower level is to determine a flow pattern that satisfies user equilibrium conditions. Another method that incorporates UE is the Path Flow Estimator (PFE) [10, 11]. The object of PFE is to find the optimized path flows. The estimated O-D flows are calculated by adding up flows on all paths connecting respective O-D pair. For the temporal part, Okutani [12] and Ashok and Ben-Akiva [13] used Kalman filter to represent the dynamic transition between consequent O-D flows. The transition equation uses an auto-regressive model to predict future O-D flows based on prior ones. The measurement equation denotes the relationship between O-D and link flows to capture the topology of transportation networks. In this paper, we use Kalman filter as a benchmark to compare the performance of our proposed approach.

The performance of traditional statistical models is limited due to their linearities. Real-time prediction requires more powerful models to represent spatial and temporal correlations. In recent years, deep neural networks have shown to be effective in approximating nonlinear features in classification, regression and control problems [14, 15]. Up to date, supervised learning plays a major part in the field of deep learning. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are effective in recognizing spatial and temporal patterns respectively [16, 17]. Since transportation networks are represented by nodes and arcs, graph-structured data appear frequently in this domain. We can incorporate prior knowledge of traffic topology into deep neural networks. Graph Neural Networks (GNNs) are shown to be effective

in dealing with graph-structured data [18]. Graph Convolutional Networks (GCNs) [19] utilize the adjacency matrix to represent node connections. Incorporating network topology into deep neural networks can accelerate convergence and improve prediction performance. Gated Graph Neural Networks (GGNNs) [20] have shown outstanding performance in time-series graph tasks. Yu et al. [21] extended GCN to time-series structure and proposed an integrated framework for spatial-temporal graph traffic forecasting. This structure can represent the evolution of node information but it can not reflect the information flow from arcs to nodes. Chen et al. [22] proposed the Line Graph Neural Networks (LGNNs) to solve the problem. However, this structure requires extensive information exchange between nodes and arcs, which can increase computing burden in practice.

In this paper, we propose the Fusion Line Graph Convolutional Networks (FL-GCNs) to forecast O-D flows. This structure consists of two parts: link flows to O-D flows and historical O-D flows to future O-D flows. In the first part, we relax some connections in LGNN and convert LGNN into the structure that can process data from link flows to O-D flows. We also incorporate the time-series information into this structure. In the second part, we use deep neural networks to represent the evolution from historical O-D flows to predicted O-D flows. We have designed Fully Connected Networks (FCNs) and Convolutional Neural Networks in this part. The proposed FL-GCN provides a general framework for processing data from links to nodes. We use real data in New Jersey Turnpike to evaluate our model. The results show that our model outperforms Kalman filter by 17.87% in predicting the whole O-D pairs.

The rest of this paper is organized as follows. In Section 2, we elaborate on our proposed model for O-D flow prediction. In Section 3, we use real traffic data to evaluate our approach and show our results and analysis. In Section 4, we summarize the conclusions and propose several directions for future work.

2 The Model

In this section, we describe the framework of Fusion Line Graph Convolutional Networks (FL-GCNs). We first introduce the preliminary definitions which would be used in our model. Then we elaborate on our model in three sections: Graph Convolutional Networks (GCNs), Line Graph Convolutional Networks (L-GCNs), and FL-GCNs. GCN is the structure that represents the evolution of link flows. L-GCN denotes the relationship from links to nodes. FL-GCN consists of two parts: link flows to Origin-Destination (O-D) flows and historical O-D flows to future O-D flows. In the first part, we use L-GCN to approximate the evolution from link flows to O-D flows. In the second part, we have designed Fully Connected Networks (FCNs) and Convolutional Neural Networks (CNNs) to represent the evolution from historical O-D flows to future O-D flows. The proposed FL-GCN provides a general framework to deal with problems related to spatial-temporal mapping from links to nodes.

2.1 Preliminary Definitions

We follow the notations in [23] and consider a directed graph $G = (V, E)$ that includes a set of nodes V and a set of links E. The network consists of n_d nodes, n_{lk} links and $n_{od} = n_d(n_d - 1)$ O-D pairs. We assume that n_l of the n_{lk} links are equipped with traffic sensors.

During an analysis period divided into equal intervals $h = 1, 2, 3, \ldots$, we use x_{rh} to represent the number of vehicles between the rth O-D pair departed in interval h. The number of traffic counts at detector l during interval h is denoted by y_{lh}. We use $\mathbf{x}_h$ to denote corresponding

$(n_d * (n_d - 1))$ vector of all O-D pairs and use $\mathbf{y}_h$ to represent corresponding $(n_l * 1)$ vector of all link flows. In addition, we use $\mathbf{x}_h^H$ and $\mathbf{y}_h^H$ to denote corresponding historical O-D flows and link flows respectively. The historical flows typically are the counts in interval h during previous days. The prediction of $\mathbf{x}_h$ is denoted by $\hat{\mathbf{x}}_h$. The O-D prediction problem can be expressed as the relationship in Eq. (1):

$$\hat{\mathbf{x}}_h = f(\mathbf{y}_{h-1}, \mathbf{y}_{h-2}, \ldots, \mathbf{y}_{h-k}, \mathbf{y}_{h-1}^H, \mathbf{y}_{h-2}^H, \ldots, \mathbf{y}_{h-k}^H, \mathbf{x}_h^H, \mathbf{x}_{h-1}^H, \ldots, \mathbf{x}_{h-m}^H), \tag{1}$$

where k and m are the number of prior intervals in link and O-D flows.

2.2 Graph Convolutional Networks (GCNs)

In our O-D prediction problem, we first use GCN to denote the evolution of link flows. Graph neural networks incorporate the traffic topology into neural networks. The GCN proposed by Kipf and Welling [19] is shown to be efficient in learning on graph-structured data.

The input and output of this structure are both link flows. To fit the structure of neural networks and incorporate time-series information, we integrate real-time link flows with historical link flows and use the $(n_l * 2k)$ vector $\mathbf{Z}_{h-1}$ to denote the integrated link flows as follows:

$$\mathbf{Z}_{h-1} = \begin{bmatrix} \mathbf{y}_{h-1} & \mathbf{y}_{h-2} & \cdots & \mathbf{y}_{h-k} & \mathbf{y}_{h-1}^H & \mathbf{y}_{h-2}^H & \cdots & \mathbf{y}_{h-k}^H \end{bmatrix}.$$

Since we consider the evolution of link flows, we use the $(n_l * n_l)$ adjacent matrix $\mathbf{A}_L$ to represent the link connections. Then we use spectral convolution to predict $\hat{\mathbf{y}}_h$, which denotes the predicted link flows in interval h. In this case, we consider the n_l link detectors as 'nodes'. Features in each 'node' include real-time link flows and historical link flows in vector $\mathbf{Z}_{h-1}$. The number of features in each 'node' is $2k$.

The spectral operation on graphs is defined as the multiplication of a signal $\mathbf{z} \in \mathbb{R}^{n_l}$ (a scalar for each node) with a filter $\mathbf{g}_\theta = \text{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^{n_l}$ in the Fourier domain:

$$\mathbf{g}_\theta \star \mathbf{z} = \mathbf{U}\mathbf{g}_\theta(\mathbf{\Lambda})\mathbf{U}^T\mathbf{z}, \tag{2}$$

where $\mathbf{U}$ is the matrix of eigenvectors of the graph Laplacian $\mathbf{L} = \mathbf{I}_{n_l} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_L\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, with a diagonal matrix of its eigenvalues $\mathbf{\Lambda}$. $\mathbf{D}$ is the degree matrix and $\mathbf{I}_{n_l}$ is the identity matrix.

The operation in Eq. (2) is computationally expensive due to the multiplication with eigenvector matrix $\mathbf{U}$, especially in large graphs. Hammond et al. [24] suggested that $\mathbf{g}_\theta(\mathbf{\Lambda})$ could be approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(x)$ with kth order. Kipf and Welling [19] limited the layer-wise convolution operation to $k = 1$ to alleviate overfitting on local neighborhood structures for graphs with very wide node degree distributions. They further approximated the largest eigenvalue of L and constrained the number of parameters. Then Eq. (2) can be approximated by:

$$\mathbf{g}_\theta \star \mathbf{z} \approx \theta \left(\mathbf{I}_{n_l} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_L\mathbf{D}^{-\frac{1}{2}} \right) \mathbf{z}.$$

Note that repeated application of the operation could lead to numerical instabilities and exploding gradients in deep neural networks. Kipf and Welling [19] introduced the renormalization trick and replaced $\mathbf{I}_{n_l} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_L\mathbf{D}^{-\frac{1}{2}}$ with $\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}_L\widetilde{\mathbf{D}}^{-\frac{1}{2}}$, in which $\widetilde{\mathbf{A}}_L = \mathbf{A}_L + \mathbf{I}_{n_l}$ and $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{L_{ij}}$. In our case, we use Random Walk Laplacian matrix $\hat{\mathbf{A}}_L = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}_L$ to replace

$\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}_L\widetilde{\mathbf{D}}^{-\frac{1}{2}}$ to simplify expressions. Then we can extend the signal $\mathbf{z} \in \mathbb{R}^{n_l}$ to $\mathbf{Z} \in \mathbb{R}^{n_l \times 2k}$ with $2k$ input features and generalize the convolution operation by:

$$\hat{\mathbf{y}} = \hat{\mathbf{A}}_L \mathbf{Z}\mathbf{\Theta},$$

where $\mathbf{\Theta} \in \mathbb{R}^{2k \times 1}$ is a matrix of filter parameters and $\hat{\mathbf{y}}$ is the $(n_l * 1)$ convolved signal matrix. We consider a two-layer GCN as follows:

$$\hat{\mathbf{y}}_h = \rho\left(\hat{\mathbf{A}}_L \sigma\left(\hat{\mathbf{A}}_L \mathbf{Z}_{h-1}\mathbf{w}_0 + \mathbf{b}_0\right)\mathbf{w}_1 + \mathbf{b}_1\right),$$

where $\rho(\cdot)$ and $\sigma(\cdot)$ are activation functions, and $\mathbf{w}_0$, $\mathbf{w}_1$, $\mathbf{b}_0$, $\mathbf{b}_1$ represent parameters in each layer.

The function of the modified adjacency matrix $\hat{\mathbf{A}}_L$ in GCN is similar to the assignment matrix in Eq. (5b). It can approximate the dynamic evolution of link flows and accelerate the convergence of deep neural networks.

2.3   Line Graph Convolutional Networks (L-GCNs)

The GCN approximates the dynamic evolution of link flows. In our O-D prediction problem, we need to represent the evolution from links E to nodes V. In this part, we transform the original directed graph $G = (V, E)$ into corresponding line graph and show the structure of L-GCN.

Consider a directed graph such as that in Fig. 2. We can transform the original graph into corresponding line graph and use the notation $L(G)$ to represent this operation. The line graph represents adjacencies between edges of G. We use an incidence matrix $\mathbf{P}$ to represent the aggregation relationship from links to nodes. Take node i in Fig. 2 for example, P_{ij} represents the link starting from node i and denotes the outflow from node i. Then we set the corresponding value in the incidence matrix to be 1. Accordingly, P_{ji} represents the inflow and the value is -1. If there is no connection between links and nodes, we should set the value in the incidence matrix to be 0.
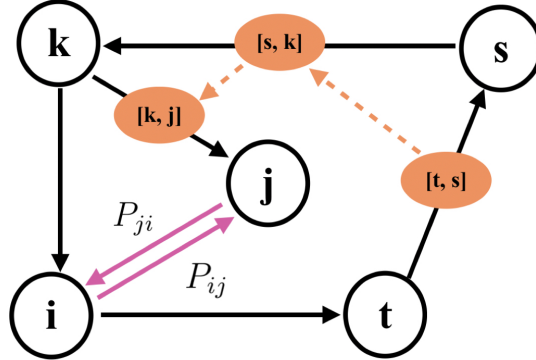
Fig. 2. Construction of the line graph and the incidence matrix.

In Fig. 2, we have the node adjacency matrix $\mathbf{A}_N$, the link adjacency matrix $\mathbf{A}_L$ and the incidence matrix $\mathbf{P}$. Chen et al. [22] proposed the Line Graph Neural Networks (LGNNs) that considered the interaction between nodes and links. LGNN includes the evolution of original graph G, the evolution of line graph $L(G)$ and the interaction between them. However, this structure requires extensive information exchange between nodes and links, which can increase

computing burden in practice. In our approach, We relax some connections in LGNN and only consider the evolution of link flows and the aggregation of link flows. The evolution of link flows is denoted by GCN and the aggregation of link flows is represented by the incidence matrix $\mathbf{P}$. The proposed L-GCN is shown as follows:

$$
\begin{aligned}
\mathbf{x}_h^L &= \phi\left(\mathbf{P}\hat{\mathbf{y}}_h\mathbf{w}_2 + \mathbf{b}_2\right) \\
&= \phi\left(\mathbf{P}\rho\left(\hat{\mathbf{A}}_L\sigma\left(\hat{\mathbf{A}}_L\mathbf{Z}_{h-1}\mathbf{w}_0 + \mathbf{b}_0\right)\mathbf{w}_1 + \mathbf{b}_1\right)\mathbf{w}_2 + \mathbf{b}_2\right),
\end{aligned}
$$

where ϕ is the activation function, $\mathbf{w}_2$ and $\mathbf{b}_2$ are parameters in neural networks. Compared with GCN, L-GCN utilizes deep neural networks to approximate the aggregation of link flows.

2.4 Fusion Line Graph Convolutional Networks (FL-GCNs)

In this part, we present the structure of FL-GCN which includes historical O-D flows. FL-GCN includes two parts: link flows to O-D flows and historical O-D flows to future O-D flows. The evolution of link flows to O-D flows is denoted by L-GCN. We can use Convolutional Neural Networks (CNNs) or Fully Connected Networks (FCNs) to represent the evolution from historical O-D flows to predicted O-D flows. The structure of FL-GCN is shown in Fig. 3.

The predicted O-D flows $\hat{\mathbf{x}}_h$ can be obtained by weighted summation of L-GCN outputs $\mathbf{x}_h^L$ and historical O-D flows $\mathbf{x}_h^H$,

$$
\hat{\mathbf{x}}_h = \psi\left(\mathbf{x}_h^L\mathbf{w_3} + \mathbf{x}_h^H\mathbf{w_4} + \mathbf{b}_3\right), \tag{3}
$$

where ψ is the activation function, $\mathbf{w_3}$ and $\mathbf{w_4}$ are weighted parameters for two branches, and $\mathbf{b}_3$ is the parameter in neural networks.
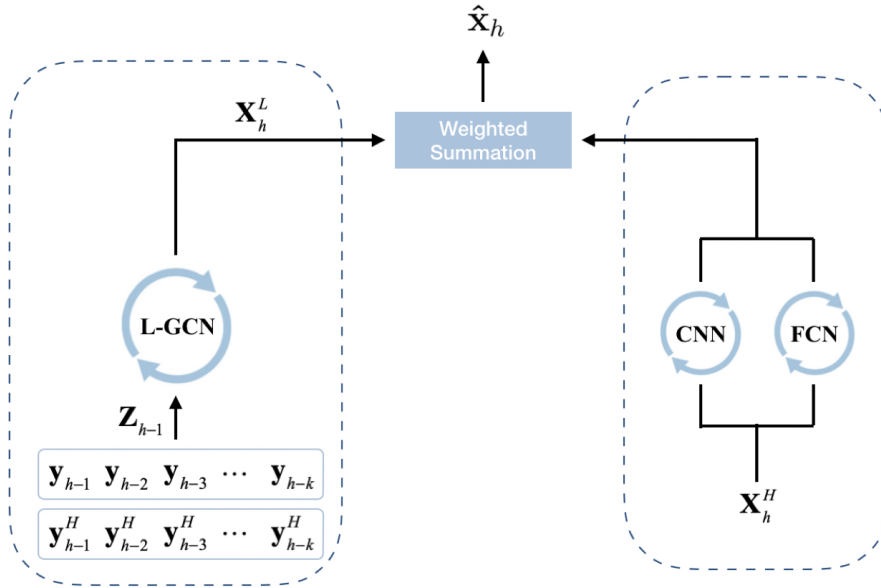
Fig. 3. The structure of FL-GCN.

The expression $\mathbf{x}_h^H\mathbf{w_4}$ in Eq. (3) is the structure of FCN, which denotes the nonlinear relationship from inputs to outputs. Note that $\mathbf{x}_h^H$ is the $(n_d * (n_d - 1))$ vector of all O-D pairs.

We can use CNN to capture adjacent O-D pair correlations. In this case, $\mathbf{x}_h^H$ is considered as an image with one channel. Then we can change the expression as follows:

$$\hat{\mathbf{x}}_h = \psi \left(\mathbf{x}_h^L \mathbf{w_3} + \mathbf{x}_h^H \circ \mathbf{w_4} + \mathbf{b}_3 \right),$$

where '$\circ$' denotes the convolution operator.

3   Case Study

New Jersey (NJ) Turnpike data were used to test the applicability of Fusion Line Graph Convolutional Networks (FL-GCNs) for Origin-Destination (O-D) prediction. Fig. 4 shows the simplified map of NJ turnpike. The dataset provides the entrance and exit times for each vehicle. We used the entrance time to calculate O-D flows. The analysis period was from 6:15 A.M. to 9:45 A.M. with the length of departure interval being 15 minutes. We assumed that the link flows were counted at the entrance of each link. To calculate the link flows, we followed the method in [23] and assumed each vehicle had a constant speed 60mph. This assumption is unrealistic in real scenarios. However, all we need is a set of consistent O-D and link flows to implement our approach. The issue of whether the constant speed consumption is reasonable is not directly relevant. Since there is only one route for each O-D pair in NJ Turnpike, the route choice effect is not considered.

There are 26 interchanges in NJ Turnpike networks. The O-D table is the $(26*25)$ matrix. We assume the entrance of each link is equipped with traffic sensors. Since there are two directions in the turnpike, the number of link counts is 50. We use O-D and link flows from February to May in 2013 to train the neural networks. Then we use O-D flows in June 2013 to evaluate our prediction performance.

When we trained the FL-GCN, 4 prior intervals (1 hour) of link flows were used to predict the next interval O-D flows. Historical O-D and link flows were the same interval 7 days ago. Since there were 50 links and the number of prior link intervals was 4, real-time and historical link flows were integrated into a $(50*8)$ matrix. We used 3 layers of Graph Convolutional Networks (GCNs) and 1 layer of line graph transformation. The adjacency matrix and incidence matrix were defined according to the topology in Fig. 4. For the historical O-D part, we used Fully Connected Networks (FCNs) and Convolutional Neural Networks (CNNs) to evaluate the spatial correlations. The number of layers in FCN and the number of layers in CNN were both 3. The kernel size of CNN was $(3*3)$.

Kalman filter was considered as a benchmark to compare the performance of FL-GCN. In Kalman filter, we followed the assumptions in [25]. Firstly, the structure of transition matrix remained constant over the whole day. Secondly, a flow between O-D pair r for a period was related only to r^{th} O-D flow of prior intervals, so the transition matrix was diagonal. The assignment matrix was calculated directly from link counts and O-D flows. In addition, we used deviations from the same interval 7 days ago as state variables [23]. The details of Kalman filter implementation are shown in Appendix A..

3.1   Results and Analysis

We use Root Mean Square Error (RMSE) and Root Mean Square Error Normalized (RMSN) in Eq. (4) to evaluate the performance. RMSE is used to measure the absolute difference and
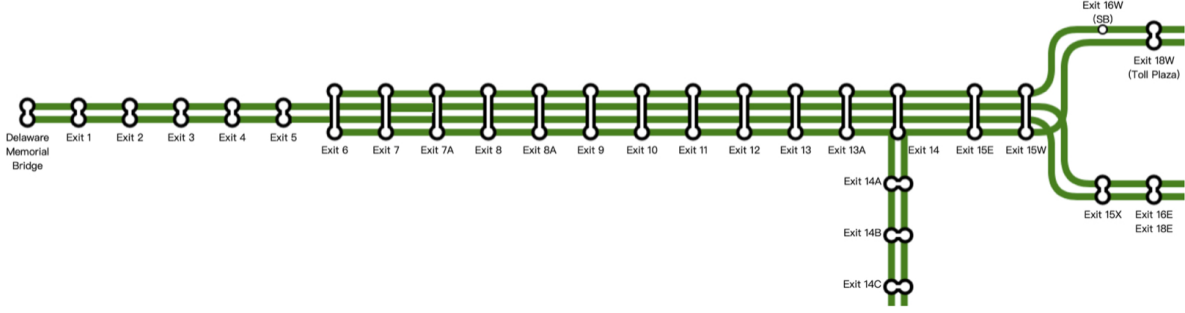
Fig. 4. Section of NJ turnpike.

RMSN is used to measure the relative difference.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)} \tag{4a}$$

$$RMSN = \frac{\sqrt{N\sum_i(x_i - \hat{x}_i)}}{\sum_i x_i}, \tag{4b}$$

where N is the total number of predicted O-D pairs in June 2013, $\hat{x}_i$ is the ith predicted O-D flow and x_i is the ground truth.

The results are shown in Table 1. We have conducted experiments with different prediction steps. FL-GCN-CNN denotes the FL-GCN approach using CNN to represent the evolution from historical O-D flows to predicted O-D flows. Accordingly, FL-GCN-FCN uses FCN to capture the evolution. The predicted performance of Kalman filter is shown as a benchmark. The final column displays the errors when historical O-D flows for each interval are used. The results show that RMSE and RMSN in FL-GCN with CNN are both smaller than those in FL-GCN with FCN, which indicates that adding CNN to capture historical O-D correlations can improve prediction performance. While the performance in FL-GCN becomes worse as we increase the prediction step, the change is not much significant. In addition, errors in two kinds of FL-GCN are both smaller than those in Kalman filter. We use FL-GCN-CNN to calculate the RMSE improvement compared with Kalman filter, the results show that FL-GCN with CNN can outperform Kalman filter by 18.02%, 17.9%, and 17.7% in predicting 1-step, 2-step, and 3-step O-D pairs respectively. The average improvement is 17.87% in predicting the whole O-D pairs.

Table 1: Prediction comparison between FL-GCN and Kalman filter.

		FL-GCN-CNN	FL-GCN-FCN	Kalman filter	Historical
	1-Step Predicted	7.629	7.967	9.383	8.445
RMSE	2-Step Predicted	7.699	7.998	9.378	8.498
	3-Step Predicted	7.747	8.015	9.413	8.554
	1-Step Predicted	0.542	0.566	0.678	0.610
RMSN	2-Step Predicted	0.547	0.568	0.670	0.607
	3-Step Predicted	0.551	0.570	0.669	0.608

Then we choose several O-D pairs to show the temporal prediction performance. The analysis period is the peak hour from 6:15 A.M. to 9:45 A.M. in June 2013. Fig. 5 shows two figures in 1-step prediction and Fig. 6 shows figures in two step and three step predictions respectively. We use FL-GCN with CNN in these cases to get better performance. From the comparison, we can see that FL-GCN can recognize the temporal patterns better than Kalman filter.
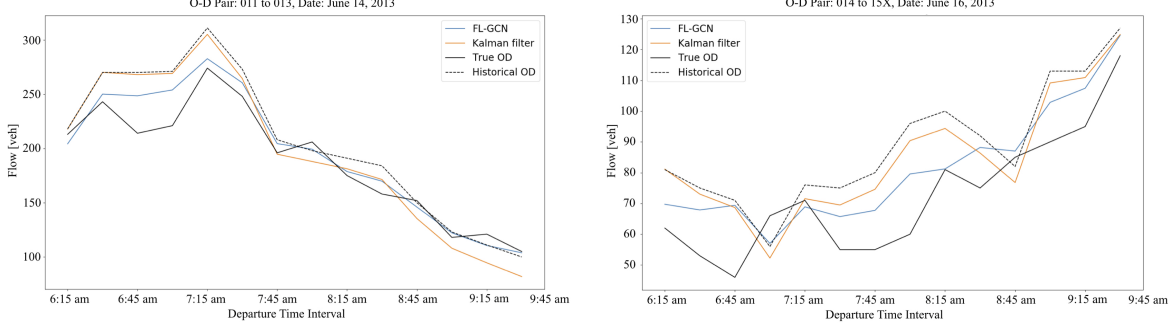
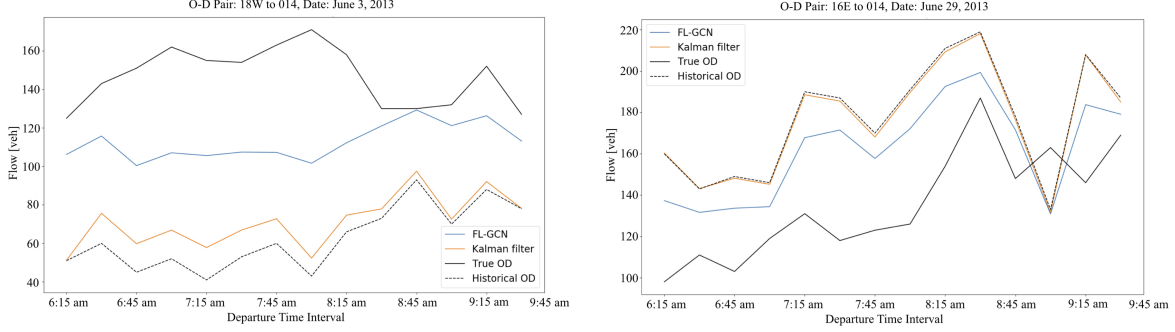**Fig. 5.** One step prediction comparison.
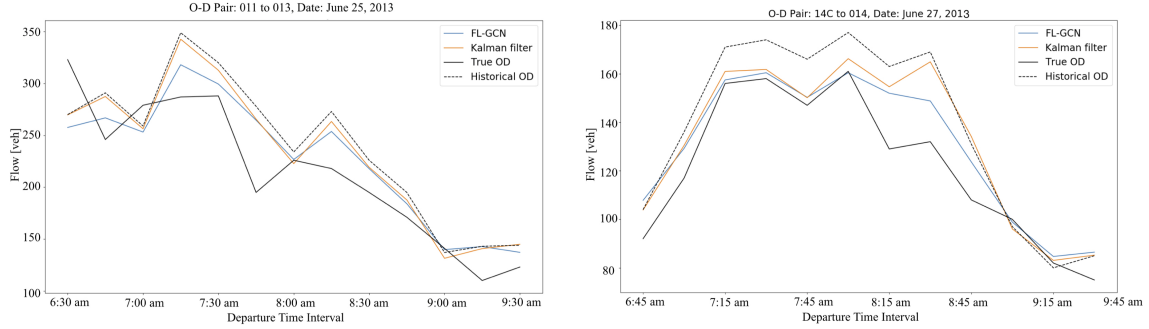
Fig. 6. Two and three step prediction comparison.

In Fig. 5 and Fig. 6, we can see that both FL-GCN and Kalman filter are predicted based on historical O-D flows and modified by link flows. Then we investigate the scenario when historical O-D flows are poor. Fig. 7 shows the 1-step prediction comparison when the gap between historical data and true O-D flow is large. The results show that FL-GCN can also have better performance. In Kalman filter, we assume that the transition matrix keeps constant during the day, which would affect the auto-regressive process in Eq. (5a). FL-GCN can use neural networks to approximate the dynamic patterns.

In Fig. 5, the first sub-figure is the O-D pair with higher flows while the second sub-figure is the one with lower flows. In Table 2, we investigate the performance when flows are less than and more than 100 vehicles per 15 minutes. The results show that Kalman filter has better performance when the flow is high. In Kalman filter, the predicted O-D flows are obtained by the summation of deviations with historical O-D flows. The time-series pattern of O-D pair with higher flow is more significant than that with lower flow. Historical O-D flows have greater effect on the final prediction results. Since FL-GCN is designed to minimize the whole prediction errors, it has better performance when the flow is less than 100 vehicles per interval.

Above all, we can conclude that FL-GCN performs better than Kalman filter for predicting the whole O-D demands, especially when we use CNN to capture historical O-D correlations.

9

Fig. 7. FL-GCN and Kalman filter prediction comparison with poor historical O-D flows.

			FL-GCN CNN	FL-GCN FCN	Kalman filter	Historical
RMSE	Flows < 100	1-Step Predicted	6.006	6.219	8.298	6.628
		2-Step Predicted	6.088	6.419	8.280	6.676
		3-Step Predicted	6.116	6.281	8.297	6.722
	Flows ≥ 100	1-Step Predicted	34.118	36.181	33.395	39.223
		2-Step Predicted	34.460	34.981	33.246	39.040
		3-Step Predicted	35.073	36.693	33.334	38.988
RMSN	Flows < 100	1-Step Predicted	0.538	0.557	0.743	0.593
		2-Step Predicted	0.539	0.568	0.733	0.591
		3-Step Predicted	0.538	0.553	0.730	0.591
	Flows ≥ 100	1-Step Predicted	0.218	0.231	0.213	0.250
		2-Step Predicted	0.222	0.225	0.214	0.251
		3-Step Predicted	0.228	0.238	0.217	0.253

Table 2: Prediction comparison between FL-GCN and Kalman filter for different flows.

Deep neural networks in FL-GCN can approximate better nonlinearity in dynamic transportation systems. This structure has potential in solving transportation problems with spatial-temporal features. Kalman filter has better performance when the O-D flow is high due to greater effect of historical O-D flows.

In addition, FL-GCN is more convenient to implement than Kalman filter. In Kalman filter, we need to prepare initial O-D flows, initial covariance matrix, transition matrix, assignment matrix, and transition errors for prediction and update steps. The predicted state vectors are deviated from historical data. We need to add historical O-D flows to get the final results. In FL-GCN, we only need to prepare O-D flows, link flows, and neural network parameters to implement the model.

4   Conclusions

This paper uses Fusion Line Graph Convolutional Networks to predict Origin-Destination (O-D) demands along a closed highway. We use New Jersey Turnpike data to evaluate our model. The results show that our model can recognize spatial and temporal patterns simultaneously. In ad-

dition, we evaluate FL-GCN with Convolutional Neutal Networks (CNNs) and Fully Connected Networks (FCNs). Since CNN can capture adjacent O-D correlations, FL-GCN with CNN has better performance than FL-GCN with FCN. Then we compare FL-GCN with Kalman filter, the results show that FL-GCN with CNN can outperform Kalman filter by 17.87% in predicting the whole O-D pairs. We also investigate the performance with different flow levels. The results show that Kalman filter has better performance when the O-D flow is high due to greater effect of historical O-D flows.

This work can be extended in several directions. First, we can use the proposed approach to deal with missing observations due to sensor failures. Second, we can extend the approach to recurrent line graph neural networks to recognize time-series patterns. Third, we can generalize the proposed framework by adding more traffic information (e.g. speed).

Acknowledgements

Appendix A.   Kalman Filter Methodology

In Kalman filter, firstly we use historical Origin-Destination (O-D) flows to estimate initial state vector $\mathbf{x_o}$, initial estimate covariance $\mathbf{p_0}$, transition matrix $\mathbf{f}$ and transition error $\mathbf{w}$. Then we use observed link flows $\mathbf{y}$, assignment matrix $\mathbf{a}$ and measurement error $\mathbf{v}$ in time interval h to predict O-D flows in interval $h + 1$.

Kalman filter includes two steps: prediction and update. We can use the state space form in Eq. (5) to represent spatial and temporal correlations. Since O-D flows in previous days contain information about spatial and temporal patterns, we use deviations of O-D flows from historical data as the state-vector. In addition, deviations can take on both positive and negative values, which would approximate a normal distribution,

$$\mathbf{x}_h - \mathbf{x}_h^H = \sum_{p=h-q'}^{h-1} \mathbf{f}_h^p(\mathbf{x}_p - \mathbf{x}_p^H) + \mathbf{w}_h \tag{5a}$$

$$\mathbf{y}_h - \mathbf{y}_h^H = \sum_{p=h-p'}^{h} \mathbf{a}_h^p(\mathbf{x}_p - \mathbf{x}_p^H) + \mathbf{v}_h, \tag{5b}$$

where $\mathbf{x}_h$ is the $(n_{od}*1)$ vector of O-D flows departing at interval h. $\mathbf{f}_h^p$ is the matrix of time-series effect of $\mathbf{x}_p$ on $\mathbf{x}_h$. $\mathbf{w_h}$ is the vector of transition errors. $\mathbf{y}_h$ is the $(n_l * 1)$ vector of link flows. $\mathbf{a}_h^p$ is the assignment matrix which denotes the relationship between O-D flows and link traffic counts. $\mathbf{v}_h$ is the vector of measurement errors. $\mathbf{x}_h^H$ and $\mathbf{y}_h^H$ are the corresponding historical O-D flows and link flows. In this case, $\mathbf{y}_h^H = \sum_{p=h-p'}^{h} \mathbf{a}_h^p \mathbf{x}_p^H$. q' is the number of lagged O-D flows which affect the hth interval flows. p' is the number of intervals taken to calculate the link flows in h^{th} interval. We assume that:
(a). $E[\mathbf{w}_h] = 0$;
(b). $E[\mathbf{v}_h] = 0$;
(c). $E[\mathbf{w}_h\mathbf{v}_j^T] = 0$;

(d). $E[\mathbf{w}_h\mathbf{w}_j^T] = \mathbf{Q}_h\delta_{hj}$, where $\delta_{hj} = 1$ if $h = j$ and 0 otherwise, $\mathbf{Q}_h$ is the covariance matrix;

(e). $E[\mathbf{v}_h\mathbf{v}_j^T] = \mathbf{R}_h\delta_{hj}$, where $\delta_{hj} = 1$ if $h = j$ and 0 otherwise, $\mathbf{R}_h$ is the covariance matrix.

Equation (5a) denotes the auto-regressive progress, which describes the temporal relationship among consequent O-D flows. Equation (5b) represents the mapping from O-D flows to link traffic counts. The assignment matrix $\mathbf{a}_h^p$ is mainly influenced by router choice behaviors and the mapping from path flows to link flows, which can represent the nonlinear topology of transportation networks. Kalman filter uses perdition and update steps to minimize the estimate covariance, which denotes the prediction uncertainty.

We use $\partial\mathbf{x_h}$ to represent the deviations, $\partial\mathbf{x}_h = \mathbf{x}_h - \mathbf{x}_h^H$. To fit the structure of Kalman filter, we use the technique of State Augmentation [23]. Let $s = max[p', q'-1]$, we re-define the state vector as follows:

$$\mathbf{X_h} = [\partial\mathbf{x}_h{}^T \quad \partial\mathbf{x}_{h-1}{}^T \quad \ldots \quad \partial\mathbf{x}_{h-s}{}^T]^T.$$

Considering the following definitions:

$$\mathbf{F_h} = \begin{bmatrix} \mathbf{f}_h^{h-1} & \mathbf{f}_h^{h-2} & \cdots & \mathbf{f}_h^{h-s} & \mathbf{f}_h^{h-(s+1)} \\ \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{W}_h = [\mathbf{w}_h{}^T \quad \mathbf{0}^T]^T.$$

Then Equation (5a) can be written as:

$$\mathbf{X}_h = \mathbf{F}_h\mathbf{X}_{h-1} + \mathbf{W}_h. \tag{6}$$

From earlier assumption, it follows that:

(a). $E[\mathbf{W}_h] = 0$;

(b). $E[\mathbf{W}_h\mathbf{W}_j^T] = \mathbf{\Theta}_h\delta_{hj}$, where $\delta_{hj} = 1$ if $h = j$ and 0 otherwise, $\mathbf{\Theta}_h$ has a top-left block $\mathbf{Q}_h$ and is zero elsewhere.

For the measurement Equation (5b), we define:

$$\mathbf{Y}_h = \mathbf{y}_h - \mathbf{y}_h^H$$

$$\mathbf{A}_h = [\mathbf{a}_h^h \quad \mathbf{a}_h^{h-1} \quad \ldots \quad \mathbf{a}_h^{h-s}].$$

Then Equation (5b) can be written as:

$$\mathbf{Y}_h = \mathbf{A}_h\mathbf{X}_h + \mathbf{v}_h. \tag{7}$$

Equation (6) and (7) together can be used to fit the Kalman filter process.

References

[1] S. Djahel, R. Doolan, G.-M. Muntean, and J. Murphy, "A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 125–151, 2015.

[2] A. M. De Souza, C. A. Brennand, R. S. Yokoyama, E. A. Donato, E. R. Madeira, and L. A. Villas, "Traffic management systems: A classification, review, challenges, and future perspectives," *International Journal of Distributed Sensor Networks*, vol. 13, no. 4, p. 1550147716683612, 2017.

[3] L. Jin, A. A. Kurzhanskiy, and S. Amin, "Throughput-improving control of highways facing stochastic perturbations," *arXiv preprint arXiv:1809.07610*, 2018.

[4] E. Cascetta, "Estimation of trip matrices from traffic counts and survey data: a generalized least squares estimator," *Transportation Research Part B: Methodological*, vol. 18, no. 4-5, pp. 289–299, 1984.

[5] M. G. Bell, "The estimation of origin-destination matrices by constrained generalised least squares," *Transportation Research Part B: Methodological*, vol. 25, no. 1, pp. 13–22, 1991.

[6] H. Spiess, "A maximum likelihood model for estimating origin-destination matrices," *Transportation Research Part B: Methodological*, vol. 21, no. 5, pp. 395–412, 1987.

[7] M. Maher, "Inferences on trip matrices from observations on link volumes: a bayesian statistical approach," *Transportation Research Part B: Methodological*, vol. 17, no. 6, pp. 435–447, 1983.

[8] E. Cascetta and S. Nguyen, "A unified framework for estimating or updating origin-destination matrices from traffic counts," *Transportation Research Part B: Methodological*, vol. 22, no. 6, pp. 437–455, 1988.

[9] H. Yang, "Heuristic algorithms for the bilevel origin-destination matrix estimation problem," *Transportation Research Part B: Methodological*, vol. 29, no. 4, pp. 231–242, 1995.

[10] H. D. Sherali, R. Sivanandan, and A. G. Hobeika, "A linear programming approach for synthesizing origin-destination trip tables from link traffic volumes," *Transportation Research Part B: Methodological*, vol. 28, no. 3, pp. 213–233, 1994.

[11] H. D. Sherali and T. Park, "Estimation of dynamic origin–destination trip tables for a general network," *Transportation Research Part B: Methodological*, vol. 35, no. 3, pp. 217–235, 2001.

[12] I. Okutani, "The kalman filtering approaches in some transportation and traffic problems," *Transportation and traffic theory*, 1987.

[13] K. Ashok and M. E. Ben-Akiva, "Estimation and prediction of time-dependent origin-destination flows with a stochastic mapping to path flows and link flows," *Transportation science*, vol. 36, no. 2, pp. 184–198, 2002.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[15] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[17] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[18] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[20] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

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

[22] Z. Chen, L. Li, and J. Bruna, "Supervised community detection with line graph neural networks," *arXiv preprint arXiv:1705.08415*, 2018.

[23] K. Ashok, "Estimation and prediction of time-dependent origin-destination flows," Ph.D. dissertation, Massachusetts Institute of Technology, 1996.

[24] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[25] K. Ashok and M. E. Ben-Akiva, "Alternative approaches for real-time estimation and prediction of time-dependent origin–destination flows," *Transportation Science*, vol. 34, no. 1, pp. 21–36, 2000.