Spatial Community-Informed Evolving Graphs for Demand Prediction *

Qianru Wang[1 [0000−0002−1682−910X]], Bin Guo ✉[1 [0000−0001−7631−3386]], Yi Ouyang[1 [0000−0001−5987−451X]], Kai Shu[2 [0000−0002−6043−1764]], Zhiwen Yu[1 [0000−0002−9905−3238]], and Huan Liu[3 [0000−0002−3264−7904]]

[1] School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, PR China
guob@nwpu.edu.cn
[2] Department of Computer Science, Illinois Institute of Technology, USA
[3] College of Computer Science and Engineering Arizona State University, USA

Abstract. The rapidly increasing number of sharing bikes has facilitated people's daily commuting significantly. However, the number of available bikes in different stations may be imbalanced due to the free check-in and check-out of users. Therefore, predicting the bike demand in each station is an important task in a city to satisfy requests in different stations. Recent works mainly focus on demand prediction in settled stations, which ignore the realistic scenarios that bike stations may be deployed or removed. To predict station-level demands with evolving new stations, we face two main challenges: (1) How to effectively capture new interactions in time-evolving station networks; (2) How to learn spatial patterns for new stations due to the limited historical data. To tackle these challenges, we propose a novel Spatial Community-informed Evolving Graphs (SCEG) framework to predict station-level demands, which considers two different grained interactions. Specifically, we learn time-evolving representation from fine-grained interactions in evolving station networks using EvolveGCN. And we design a Bi-grained Graph Convolutional Network(B-GCN) to learn community-informed representation from coarse-grained interactions between communities of stations. Experimental results on real-world datasets demonstrate the effectiveness of SCEG on demand prediction for both new and settled stations. Our code is available at https://github.com/RoeyW/Bikes-SCEG

Keywords: Spatial-temporal analysis · Urban computing · Demand prediction · Graph Neural Network

1 Introduction

Sharing-bike is becoming an increasingly popular means for commuting due to its cheap cost, easy access and convenient usage. Bike riders can check-in to ride

* This work was partially supported by the National Key R&D Program of China(2019YFB1703901), the National Natural Science Foundation of China (No. 61772428,61725205,61902320,61972319) and China Scholarship Council.

the bike from one station and check-out at any other stations. According to the reports of Citi Bike, the largest bike-sharing company in New York City, the number of total bikes grew up to 12,000 and over 800 stations were added in 2017. Also, it is estimated that bike riders take 10 million trips in one year[1]. Due to free check-in and check-out demands of bike rides, available bikes in some stations are shortage while available bikes in others are redundant. The bike company needs to balance bikes in stations manually according to demands of stations. Therefore, demand prediction is important to balance bikes for the stations in a city.

The majority of recent works on sharing-bike demand prediction focus on settled stations, which are all existing during training and test phases, using geographical distance and temporal correlations between stations. For example, Lin *et al.* [4] used a data-driven method to learn the relationship between stations with graph neural networks to predict future demands. Chai *et al.* [9] fused multiple graphs constructed from the perspectives of the spatial distance, trip records and check-in/check-out correlations between settled stations to predict bike's demands. They delete the new stations from the dataset, which only appear few days. In the real-world scenario, new stations may be added or removed by the bike-sharing company to better satisfy users' demands. For example, according to the report of Citi Bike, the bike company added over 100 stations in one year[1]. At the early stage of adding a new station, it has limited historical demands and interactions (trips to/from other stations). But existing works may not be directly applied to predict demands for new stations due to relying on long term historical data. Therefore, it is necessary to propose a more effective model to predict demands for stations when new stations are added.

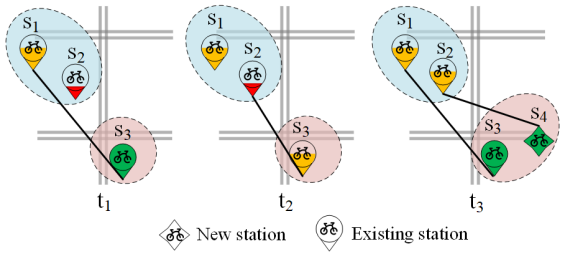


Fig. 1: The Illustration of time evolving station networks. The color of stations changes from green to red when the number of available bikes decrease. The lines between stations indicate that there are riding trips between them. The blue and red dash circles indicate communities of stations.

However, it is a non-trivial task to perform prediction when new stations are added. We face two main challenges to deal with evolving new stations. From the **temporal** aspects, it is necessary to capture new interactions, which continuously appear especially when new stations are added. Besides changing

[1] https://www.citibikenyc.com/about

of interactions between settled stations, new interactions between new stations and settled stations may appear over time, which affect demands of new stations and settled stations. As shown in Fig.1, the demands of stations are different due to the different interactions at each timestamp. For example, users would like to ride from S_1 to S_3 at t_1. And some users chose S_2 to start the trips at t_2. Specifically, there would be some new interactions that haven't appeared before a new station was added. For example, S_4 was added at t_3. Then the interactions between S_2 and S_4 appeared at that time, which will affect the demands of S_2 and S_4. Therefore, to predict demands of stations, we need to consider dynamic interactions over time, especially for new interactions.

From the **spatial** aspects, spatial interaction patterns of new stations are difficult to be learned due to the limited historical interactions, which make it difficult to predict the demands of new stations. A straightforward way to provide information for the new station is using spatial interactions of the nearby stations directly. However, station-level interactions exist fluctuation and random [23], which cannot learn the interaction patterns of new station accurately. Compared to station-level interactions, stations within a spatial community have similar demand trends and more distinct interaction patterns as observed. Thus, when a new station is added into a spatial community, we can use the historical information of its community to supply demand trends and possible interactions for this new station. For example, stations in the blue community usually interact with stations in the red community as shown in Fig 1. When the new station S_4 is added into the red community, S_4 may interact with stations in the blue community. Therefore, it is necessary to consider possible communities that new stations will interact with based on community-level interactions .

To tackle the aforementioned challenges, we consider two different-grained spatial interactions in a city. The fine-grained interactions are trips between stations, which provide specific interactions at each timestamp. The coarse-grained interactions are trips between communities of stations, which have more steady distributions in a city. Therefore, we can predict future demands by extracting the fine-grained interactions while relying on the coarse-grained interactions. To this end, we propose a novel framework which exploits Spatial Community-informed Evolving Graphs (SCEG) to predict demands with evolving new stations in station networks. Firstly, we adopt EvolveGCN [5] to represent the dynamic interactions in each timestamp to handle time-evolving station networks, which is called as *Time-evolving representation*. To deal with different-grained spatial graphs, we then design a Bi-grained Graph Convolution Network (B-GCN) to represent station-level representation based on community-level interactions, which is called as *Community-informed representation*. Finally, we use variational autoencoder [11] to fuse two kind of representations while considering some variances in the real world. In summary, the main contributions of our paper are as follows:

1. We consider a novel scenario for demand prediction that station networks would involve some new stations.

2. We propose a novel framework (SCEG), which exploits spatial community informed time-evolving graphs to predict demands for settled stations and new stations.
3. We design a Bi-grained Graph Convolutional Network (B-GCN), which assigns community-level interactions to stations.
4. We conduct experiments on the real-world datasets in New York City and Washington D.C.. Experimental results show that our model outperforms existing state-of-the-art baselines both on settled and new stations, especially when new stations are more than settled stations.

2 Problem Statement

To better represent interactions, we use graphs to model the relationship between communities or stations. To represent coarse-grained interactions between spatial communities and fine-grained interactions between stations, we use three kinds of graphs to define the problem: time-evolving station graphs, a spatial communities graph and bi-grained graphs.

Definition 1. *Time-evolving station graphs* $G^S = (G_1^S, ..., G_T^S)$.

At the t^{th} timestamp, $G_t^S = (S_t, E_t^S)$, $S_t = (s_1, ..., s_i, ...)$ is the station set in a city. $e_t^{ij} \in E_t^S$ represents the number of interactions between station s_i and s_j. A_t^S denotes the binary adjacent matrix.

Definition 2. *A spatial community graph* $G^C = (C, E^{Com})$ *is constructed to indicate the interactions between spatial communities, which is a weighted graph.*

We denote C and E^{Com} as the set of communities and edges between communities. Each community consists of stations with similar attributes (such as spatial distance, demands on weekdays). A^C is a weighted adjacent matrix. $a_{ij}^C \in A^C$ represents the probability of riding from one community to another, which is calculated by the number of trips between c_i and c_j in the number of trips from/to c_i (i.e., $a_{ij}^C = \frac{\#trips(c_i,c_j)}{\#trips(c_i)}$).

Definition 3. *Bi-grained graphs* $G^B = (G_1^B, ..., G_T^B)$ *is used to link time-evolving station graph* G_t^S *and the spatial community graph* G^C.

At the t^{th} timestamp, $G_t^B = (S_t, C, E_t^B)$, $E_t^B \in \mathbb{R}^{|S_t| \times |C|}$ represents edges between stations and communities. $a_{t,ij}^B \in A_t^B$ is equal to 1 if the i^{th} station belongs to the j^{th} community.

Additionally, we use $D_t = \{D_t^i, i \in \{in, out\}\}$ as check-in and check-out demands of stations. F^t denotes the external temporal features, e.g., temperature, wind speed, weather and weekday. These temporal features significantly affect the demand of stations. For example, demands on rainy days are less than sunny days. We give the formal problem definition as follows:

Given G^C, $G_{t-\Delta t}^B$ and a Δt period of historical data set $X^E = \{G_{t-\Delta t:t-1}^S, D_{t-\Delta t:t-1}, F_{t-\Delta t:t-1}\}$, we predict the check-in and check-out demand D_t for stations at t^{th} timestamp.

3 The Proposed Model

To predict station-level demands, we proposed SCEG to exploit time-evolving station networks informed by spatial community. As shown in Fig 2, our model consists of two phases: *learning phase* and *prediction phase*. In the learning phase, we learn time-evolving representation and community-informed representation of stations from the dynamic interactions of station graphs and interactions between spatial communities. In the prediction phase, we predict future demands conditioned on these two latent representations.

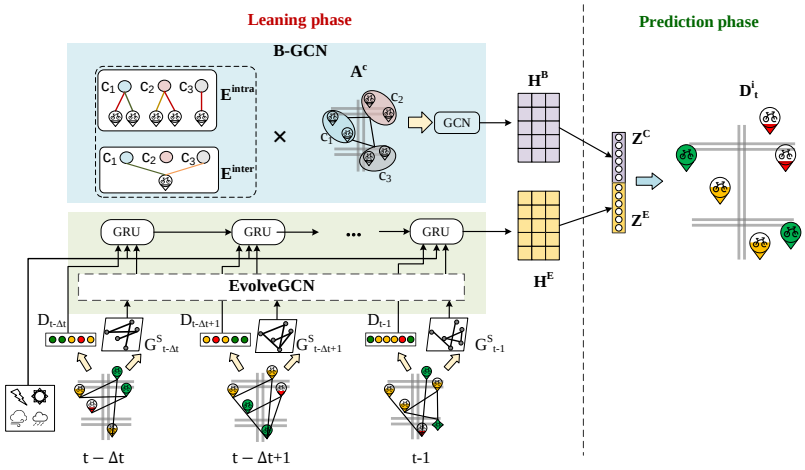


Fig. 2: The architecture of SCEG model. The learning phase is used to infer latent representations of spatial communities and time-evolving graphs. The prediction phase is used to predict future demands conditioned on these two latent representations

Learning phase is used to encode the spatial community graph and time-evolving information into two latent representations: *time-evolving representation* Z^E and *community-informed representation* Z^C separately. Since the true posterior distributions $p(Z^E|X^E)$ and $p(Z^C|G^C, G^B_{t-\Delta t})$ are intractable, the inference model $q_\phi(Z^E|X^E)$ and $q_\phi(Z^C|G^C, G^B_{t-\Delta t})$ are introduced to approximate them, where $q_\phi(Z^E|X^E) = \mathcal{N}(\mu^E, (\sigma^E)^2 I)$, $q_\phi(Z^C|G^C, G^B_{t-\Delta t}) = \mathcal{N}(\mu^C, (\sigma^C)^2 I)$. ϕ represents encoder parameters $\{\mu^E, \sigma^E, \mu^C, \sigma^C\}$.

Prediction phase is used to predict future demand based on the fusion of the two latent representations. The future check-in/out demand distribution $p(D^i_t)$ is written as:

$$\log p(D^i_t) = \log \iint p(D^i_t|Z^E, Z^C) p(Z^E) p(Z^C) d_{Z^E} d_{Z^C} \tag{1}$$

3.1 Learning phase

Time-evolving graphs representation To capture evolving patterns for stations at each timestamp, existing works used a single GCN to encode station graphs in each timestamp, which shared parameters among the various timestamps. However, a single GCN for graphs at each timestamp is hardly to predict demands of new stations. Therefore, we need to learn interactions between settled stations and new stations in time-evolving station graphs. Besides time-evolving station graphs, there are some temporal features (e.g., temperature, wind speed, demands of stations), which also contribute to demands of stations. We need to combine these temporal features while encoding the time-evolving graphs.

Firstly, we use EvolveGCN to encode changes of nodes and edges for the time-evolving graphs. We use GRU (gated recurrent unit) [13] to update the hidden state on $t-1$ timestamp as shown in Eq.(2).

$$\begin{aligned} W_t^{(l)} &= GRU_1(W_{t-1}^{(l)}) \\ H_t^{(l)} &= ReLU(A_t^S \times H_t^{(l-1)} \times W_t^{(l)}) \end{aligned} \tag{2}$$

where $W_t^{(l)}$ and $H_t^{(l)}$ represent a weight matrix and a hidden state for the l^{th} layer at timestamp t. $ReLU(\cdot)$ is a rectified linear unit. The symbol $\times$ means matrix multiplication.

Secondly, we combine the embedding of time-evolving graphs with temporal features. To obtain temporal embedding of the temporal information, we use another GRU to encode them as shown in Eq.(3).

$$\widetilde{H}_t = GRU_2(H_{t-1}^{(l)}, D_{t-1}, F_{t-1}) \tag{3}$$

Based on the embedding of temporal information(X^E), we can approximate the distribution $Z^E \sim q_\phi(Z^E|X^E)$ by estimating μ^E and σ^E as shown in Eq.(4). Due to some random between station-level interaction, we sample Z^E using a random normal distribution ϵ_1.

$$\begin{aligned} H^E &= \tanh(\widetilde{H}_t \times W^E + b^E) \\ \mu^E &= FC_1(H^E), \quad \log(\sigma^E)^2 = FC_2(H^E) \\ Z^E &= \mu^E + \sigma^E \cdot \epsilon_1, \quad where \quad \epsilon_1 \sim \mathcal{N}(0, I) \end{aligned} \tag{4}$$

where W^E, b^E are trainable variables. $FC(\cdot)$ denotes fully connected layers. The symbol $\cdot$ means element-wise multiplication.

Bi-grained graph representation Due to frequently changing of time-evolving graphs, we involve a steady and coarse-grained graph, also called as a spatial community graph, which also helps infer demand trends and possible interactions of new stations. Based on this idea, we need to assign the interactions of one cluster to the stations in it. As we all know, stations' demands are different even though they have similar demand trends in the same community. Therefore, we should consider the differences between stations in a community when

assigning the community's information. On the other hand, each station also interacts with other communities. So it is necessary to explore how a station is affected by other communities.

To find spatial communities in a city, we group stations in a long period (i.e., 6 months) by K-Means, which uses Euclidean distance and their demands on different weekdays as features. After the period, we add a new station into a spatial community according to the distance and its few historical demands.

To address issues above, we firstly designed a Bi-grained Graph Convolution Network (B-GCN), which leverages a bi-grained graph to learn Community-informed representation. Specifically, we use a single GCN to represent interactions between spatial communities.

$$E^C = ReLU(A^C \times H^C \times W^C) \tag{5}$$

where H^C and W^C are trainable variables.

To transfer the embedding of community-level interactions to station-level, we use a bi-grained graph ($G^B_{t-\Delta t}$) to weight the affect of the community stations' belong to and other communities separately. Intra-weight represents how we assign the information of a community to stations in it as shown in Eq.(6). Eq.(7) represents inter-weight, which calculates how a station interacts with other communities.

$$e^{intra}_{t,ij} = \begin{cases} \frac{w^{intra}_{ij} \cdot a^B_{t-\Delta t,ij}}{\sqrt{\sum_{i|s_i \in c_j} (w^{intra}_{ij} \cdot a^B_{t-\Delta t,ij})^2}}, & s_i \in c_j \\ 0, & s_i \notin c_j \end{cases} \tag{6}$$

$$e^{inter}_{t,ij} = \begin{cases} \frac{w^{inter}_{ij} \cdot (1 - a^B_{t-\Delta t,ij})}{\sqrt{\sum_{j|s_i \notin c_j} (w^{inter}_{ij} \cdot (1 - a^B_{t-\Delta t,ij}))^2}}, & s_i \notin c_j \\ 0, & s_i \in c_j \end{cases} \tag{7}$$

where $w^{intra}_{ij} \in W^{intra}$, $w^{inter}_{ij} \in W^{inter}$,$e^{intra}_{t,ij} \in E^{intra}_t$, $e^{inter}_{t,ij} \in E^{inter}_t$. And $W^{intra}, W^{inter}, E^{intra}_t, E^{inter}_t \in \mathbb{R}^{|S_{t-\Delta t}| \times |C|}$.

Using intra-weight and inter-weight, we calculate stations' representations H^B which is transferred from communities' representations.

$$H^B = (E^{intra}_t + E^{inter}_t) \times E^C \tag{8}$$

Secondly, we learn the latent community-informed representation. Though interactions between spatial communities seem regular than those between stations, they still have some randomness. Therefore, we need to consider some randomness based on $q_\phi(Z^C|G^C, G^B_{t-\Delta t})$ we learned. To obtain the latent variable $Z^C \sim q_\phi(Z^C|G^C, G^B_{t-\Delta t})$, we estimate μ^C and σ^C of $q_\phi(Z^C|G^C, G^B_{t-\Delta t})$ as follows:

$$\begin{aligned} &\mu^C = FC_3(H^B), \quad \log(\sigma^C)^2 = FC_4(H^B) \\ &Z^C = \mu^C + \sigma^C \cdot \epsilon_2 \quad where \quad \epsilon_2 \sim \mathcal{N}(0, I) \end{aligned} \tag{9}$$

where W^C is a weight matrix, Z^C is sampled from $q_\phi(Z^C|G^C, G^B_{t-\Delta t})$ using a random normal distribution ϵ_2.

3.2 Prediction phase

Future demand is affected by the fusion of spatial community and time-evolving information. We need to consider specific information in time-evolving information while relying on coarse-grained distribution in a city. Therefore, conditioned on two independent latent variables Z^C and Z^E, we estimate the future demand distribution $p_\theta(D_t^i|Z^C, Z^E)$ as follows:

$$p_\theta(D_t^i|Z^C, Z^E) = FC_5(Z) \tag{10}$$

where Z is concatenation of $[Z^E, Z^C]$. θ is a set of trainable variables in $FC_5(\cdot)$.

To learn the encoder and decoder parameters (ϕ and θ), we need to maximize the lower bound of $p(D_t^i)$ as shown in Eq.(11), which is derived from Eq.(1):

$$\begin{aligned}
&\log p(D_t^i) \\
&\geq \iint q_\phi(Z^C, Z^E|G^C, G^B_{t-\Delta t}, X^E) \log \frac{p_\theta(D_t^i, Z^C, Z^E)}{q_\phi(Z^C, Z^E|G^C, G^B_{t-\Delta t}, X^E)} d_{Z^C} d_{Z^E} \\
&= \mathbb{E}_{q_\phi(Z^C, Z^E|G^C, G^B_{t-\Delta t}, X^E)} \left[p_\theta(D_t^i|Z^C, Z^E) \right] \\
&- D_{KL}\left(q_\phi(Z^C|G^C, G^B_{t-\Delta t}) \parallel p(Z^C) \right) \\
&- D_{KL}\left(q_\phi(Z^E|X^E) \parallel p(Z^E) \right) \triangleq \mathcal{L}(D_t^i; \phi, \theta)
\end{aligned} \tag{11}$$

where $p(Z^C) = p(Z^E) = \mathcal{N}(0, I)$.

Meanwhile, we also need to measure how accurately the model predicts future demand. We use a mean squared error as a loss function to measure the prediction error:

$$\mathcal{L}_{mse} = \frac{1}{n}(\widetilde{D}_t^i - D_t^i)^2 \tag{12}$$

Therefore, we need to optimize the loss function as follows to train the model:

$$\mathcal{L} \backsimeq \mathcal{L}_{mse} + \alpha \left[D_{KL}\left(q_\phi(Z^C|G^C, G^B_{t-\Delta t}) \parallel p(Z^C) \right) + D_{KL}\left(q_\phi(Z^E|X^E) \parallel p(Z^E) \right) \right] \tag{13}$$

4 Experiments

In this section, we conduct several experiments to evaluate our model on settled stations and new stations.

4.1 Datasets

We evaluate our model on two public real-world datasets as shown in Table 1. Bike-sharing datasets in New York City and Washington D.C. are collected from Citi Bike website[2] and Capital Bike website[3], respectively. A station, which

[2] https://www.citibikenyc.com/system-data
[3] https://www.capitalbikeshare.com/system-data

starts to be checked in/out from a certain timestamp, is defined as a new station. Compared to the dataset in New York City, new stations are more than settled stations in Washington D.C.. Besides the bike-sharing data, we use some external temporal data for two cities which are used to characterize urban dynamics, such as meteorology data (e.g., weather type, wind speed, temperature)[4][5] and holiday data (e.g., workday, holiday)[6]. Temperature and wind speed are normalized as continuous variables which range from 0 to 1. Other temporal data is encoded to one-hot variables. We choose the data of last 14 days as test set in each city. 85% of the remained data are training set and 15% of the remained data are validation set.

Table 1: The statistics of the dataset.

Data	NYC	Washington D.C.
Duration	16/10/1–17/10/27	11/01/01–12/12/9
Records	17,726,635	3,166,051
# stations	846	193
# new stations	259	105

4.2 Experiment settings

Experimental setup We predict daily demand for each station. To predict the daily demand, we use a sliding window to get the historical data. For new stations, we mask their data before they appear. We use one GCN layer, one EvolveGCN layer and one GRU layer on both datasets. We adopt different hidden units of layers in different cities. For dataset in New York City, we cluster the stations into 20 communities. We set 64 hidden units for the GCN layer in B-GCN, 128 hidden units for the EvolveGCN layer and 128 hidden units for the GRU_2 layer. FC_1 and FC_2 are set as 128 hidden units. The learning rate is set as 5×10^{-4}. For dataset in Washington D.C., we cluster the stations into 20 communities. We separately use 32, 64, 128 hidden units for the GCN layer, EvolveGCN layer and GRU_2 layer. And FC_3 and FC_4 are 64 hidden units. The learning rate is also set as 5×10^{-4}.

Baselines We compare our model (SCEG) with the following state-of-the-art works on demand prediction:

- **GRU** [13]: it only uses historical demand, meteorology data and holiday data as input.
- **T-GCN** [4,6]: They use a single GCN to encode the time-evolving station graphs into spatial embedding in each timestamp, then use GRU to encode the temporal features and spatial embedding over time.
- **E-GCN** [5]: It uses EvolveGCN to encode spatial information in each timestamp, then uses GRU to encode the temporal features and spatial embedding.

[4] https://www.kaggle.com/selfishgene/historical-hourly-weather-data
[5] https://www.kaggle.com/marklvl/bike-sharing-dataset
[6] https://www.opm.gov/policy-data-oversight/pay-leave/federal-holidays

– **Multi-graph** [9]: It uses multiple graphs including distance graph, correlation graph, and interaction graph as input of one timestamp, then uses LSTM to encode information in each timestamp.

In addition to the above state-of-the-art methods, we provide another two variants related to our model:

– **CT-GCN**: Based on T-GCN, it uses another GCN to encode the spatial community graph. The embedding of the time-evolving information and spatial communities are concatenated to predict demands.
– **SCEG-w/oBI**: It directly flattens community embedding after GCN when calculating community-informed representation, which does not use B-GCN to assign community interactions to stations.

Evaluation metrics We use two evaluation metrics: mean absolute percentage error (MAPE) and root mean square percentage error (RMSPE) similar to [14].

$$MAPE^* = \frac{1}{t \times n_t^*} \sum_{t,n_t^*} \frac{\left|\widetilde{D}_t^{i*} - D_t^{i*}\right|}{D_t^{i*}} \quad RMSPE^* = \frac{1}{t} \sum_t \sqrt{\frac{1}{n_t^*} \sum_{n_t^*} \left(\frac{\widetilde{D}_t^{i*} - D_t^{i*}}{D_t^{i*}}\right)^2} \tag{14}$$

where $\widetilde{D}_t^i$ is prediction demand and D_t^i is the ground-truth demand. n_t^* is the number of stations at t^{th} timestamp.

We evaluate the results in three perspectives: performance on all stations in t^{th} timestamp ($MAPE^{all}$, $RMSPE^{all}$), performance on settled stations ($MAPE^{settled}$, $RMSPE^{settled}$) and performance on new stations ($MAPE^{new}$, $RMSPE^{new}$).

4.3 Prediction results

We evaluate check-in and check-out demands prediction with 6 baselines as shown in Table 2. Values in bold represent the best performance. The results show that the overall performance of our model is better than other baselines. our model performs much better than other baselines when the number of new stations is more than the number of settled stations.

We first predict demand using GRU on both datasets, which only uses historical demands and external temporal data. Compared to GRU, other methods perform better as a result of considering the interaction between stations. It infers that station-level demand prediction is significantly related to spatial interactions.

Compare with T-GCN, CT-GCN performs a little better as it consider the spatial community. It infers that the steady and coarse-grained graph helps reduce some prediction errors. But it still performs worse than E-GCN, which doesn't use the embedding of spatial community. Due to using a single GCN, T-GCN and CT-GCN cannot characterize the time-evolving interactions between stations. Though multi-graph performs better than T-GCN and CT-GCN by characterizing the relationship between stations from different views, it still perform worse than E-GCN as it doesn't have a time-evolving structure. Therefore,

Table 2: Prediction errors of check-in and check-out demands.

City	Method	Check-in			
		$MAPE^{all}$	$RMSPE^{all}$	$MAPE^{settled}$	$RMSPE^{settled}$
N.Y.C.	GRU	0.611	2.144	0.407	0.848
	T-GCN	0.582	2.098	0.393	0.791
	CT-GCN	0.496	1.425	0.349	0.721
	E-GCN	0.446	1.323	<u>0.304</u>	<u>0.644</u>
	Multi-graph	0.460	1.100	0.366	0.772
	SCEG-w/oBI	<u>0.426</u>	<u>1.068</u>	0.345	0.726
	SCEG	**0.383**	**0.969**	**0.271**	**0.591**
W.D.C.	GRU	0.936	1.894	1.083	1.910
	T-GCN	0.699	1.411	0.679	1.225
	CT-GCN	0.602	1.104	0.650	1.408
	E-GCN	0.583	1.026	0.550	1.086
	Multi-graph	0.515	0.970	0.545	1.045
	SCEG-w/oBI	<u>0.508</u>	<u>0.949</u>	<u>0.445</u>	<u>0.795</u>
	SECG	**0.453**	**0.899**	**0.437**	**0.763**

City	Method	Check-out			
		$MAPE^{all}$	$RMSPE^{all}$	$MAPE^{settled}$	$RMSPE^{settled}$
N.Y.C.	GRU	0.888	3.381	0.704	2.114
	T-GCN	0.694	1.973	0.706	2.010
	CT-GCN	0.533	1.689	0.460	1.359
	E-GCN	0.494	1.435	<u>0.307</u>	<u>0.602</u>
	Multi-graph	0.511	1.310	0.355	0.773
	SCEG-w/oBI	<u>0.471</u>	<u>0.959</u>	0.437	0.850
	SCEG	**0.357**	**0.825**	**0.256**	**0.488**
W.D.C.	GRU	0.750	1.340	0.719	1.316
	T-GCN	0.649	1.118	0.678	1.234
	CT-GCN	0.646	1.148	0.629	1.103
	E-GCN	0.586	1.042	0.605	1.004
	Multi-graph	0.559	1.016	0.566	1.092
	SCEG-w/oBI	<u>0.516</u>	<u>0.998</u>	<u>0.561</u>	<u>1.088</u>
	SECG	**0.491**	**0.945**	**0.449**	**0.833**

– $MAPE^{all}$ and $RMSPE^{all}$ are metrics to evaluate performance on all stations. $MAPE^{settled}$ and $RMSPE^{settled}$ are metrics to evaluate performance on settled stations.
– Numbers in bold denote the best performance and numbers with underlines denote the second best performance.

capturing time-evolving interactions accurately plays an important role in prediction. Compared to E-GCN, SCEG-w/oBI performs better on W.D.C dataset, whose new stations are more than settled stations. It infers that E-GCN do well in encoding dynamic interactions between settled stations. But E-GCN ignores the spatial community, which doesn't help prediction for new stations with patterns of the whole city. And the latent representation combining spatial community and time-evolving graphs help infer possible interactions. Meanwhile, SCEG-w/oBI considers some random when models latent representations, which is more suitable for real-word situations. Compared to SCEG-w/oBI, SCEG uses B-GCN, which relates stations and communities reasonably, which improves the performance significantly.

In summary, the overall performance of our model is better than other baselines. And our model significantly improves overall performance when new stations are more than settled stations.

4.4 Capability of predicting new stations

To evaluate the performance on the scenario of new stations, we calculate $MAPE$ and $RMSPE$ of new stations. Meanwhile, we illustrate changes of prediction errors for a new station with different lengths of historical data.

Table 3: Prediction errors for new stations

City	Method	$MAPE^{new}$	$RMSPE^{new}$
NYC	GRU	2.380	6.663
	T-GCN	1.193	2.760
	CT-GCN	1.189	2.648
	E-GCN	1.184	2.570
	Multi-graph	1.256	3.020
	SCEG-w/oBI	<u>0.907</u>	<u>1.329</u>
	SCEG	**0.675**	**1.275**
W.D.C.	GRU	0.976	1.582
	T-GCN	0.865	1.397
	CT-GCN	0.838	1.361
	E-GCN	0.636	1.145
	Multi-graph	0.643	1.150
	SCEG-w/oBI	<u>0.581</u>	<u>1.110</u>
	SCEG	**0.530**	**0.996**

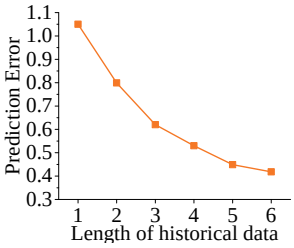


Fig. 3: Evaluation on different lengths of historical data

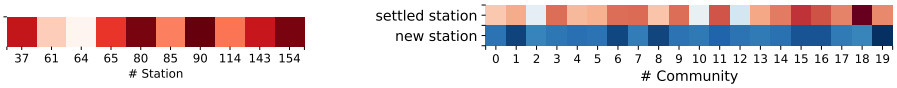
Prediction errors of new stations We show the prediction results of check-out demands for new stations in Table 3. The results show that our model performs better than other baselines on new stations. As expected, GRU and T-GCN have poor performance on handling new stations, which are not trained before. Though CT-GCN improves a little performance by involving spatial community, it cannot learn dynamic information from time-evolving graphs. Compared with CT-GCN, E-GCN performs better. It reveals that the spatial community without some time-evolving information will not predict demands of new stations well, especially in the situation that there are more new stations added. SCEG-w/oBI improve performance obviously as the latent spatial representation is helpful to predict future interactions and demands of new stations. And adding some random is useful for new stations whose spatial patterns is unknown. SCEG weights the affect of communities on each stations, which contribute to analyze the trend of new stations. Therefore, SCEG outperforms than other baselines by using time-evolving representation and community-informed representation. We decrease at least 23.2% on the dataset of New York City and 5.1% on the dataset of Washington D.C.in terms of $MAPE$. Compared to Multi-graph which is also a demand prediction work, we decrease 58.1% on dataset of N.Y.C. and 11.3% on dataset of W.D.C. in terms of $MAPE$.

Evaluation on different lengths of historical data We select one new station to find the changes in prediction errors over the length of historical data. The station was deployed on 2012/11/29 in Washington D.C., which was not trained before. We use different lengths of historical data, which range from 1 day to 6 days. As shown in Fig.3, the prediction error decreases obviously when we involve more historical data. However, the prediction error decreases slowly

when the length of historical data reaches 6 days. The reason may be that distant data cannot provide a lot of relevant information. It infers that involving more than 6 days' data cannot help a lot that will cost more time to predict demands. So we select 6 days' historical data, considering a balance between performance and time cost.

4.5 Visualization of intra-weights and inter-weights

To analyze the affect of communities on stations, we demonstrate the visualization of intra-weights and inter-weights on 2012/11/29 as shown in Fig.4. Fig.4(a) shows the intra-weights in Community #5, which consists of 10 stations. From the result, we learn that occupation rates of stations are different. Station #154, #90 and #80 are popular stations in this community. And Station #64 and #61 have fewer demands. Fig.4(b) shows the inter-weights of the settled stations and the new stations. The settled station belongs to Community #12 and the new station belongs to Community #10. The result shows that the settled station frequently interacts with Community #18. The new station rarely interacts with Community #19, Community #6 and Community #8. From the result, we learn that the new station has much fewer interactions than the settled stations. But it's still obvious that a station has different interactions with communities even though the new station has few interactions currently.



(a) Intra-weights in Community #5. The square which is more dark denotes more occupation rate.

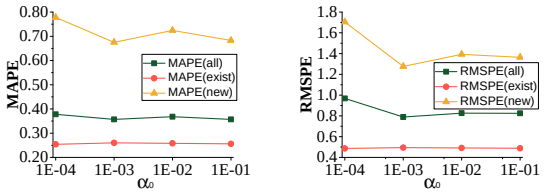
(b) Inter-weights for the settled station and the new station. Squares' colors change from dark blue to dark red mean the weights change from low to high.

Fig. 4: Visualization of intra-weights and inter-weights

4.6 Parameter analysis

In this part, we evaluate α in the loss function. Because of the difficulty of training VAE, we train our model by warming up. α is changed in each epoch to balance $\mathcal{L}_{mse}$ and KL divergences ($\alpha = epoch * \alpha_0$). To evaluate the effect of different α, we choose the different α_0 at the initial stage which ranges from 1×10^{-4} to 1×10^{-1} and train the model with the same number of epochs. The results shown in Fig. 5 are prediction errors of check-in demands in New York City. With increasing of α_0, the prediction errors of new stations are significantly affected. But prediction errors of all stations and settled stations are little affected. The prediction errors of new stations (yellow line) reach the lowest when α_0 is equal to 1×10^{-3}. When α_0 is equal to 1×10^{-3}, the model reaches better performance and consumes as much time as other parameter settings. The results illustrate

that the model relies more on $\mathcal{L}_{mse}$ at the beginning of training, which help learn the representations of spatial demands. Due to few historical data, demands of new stations will be predicted better after learning the representation of spatial demands well. However, it is difficult to learn the latent spatial representation directly, we should gradually adjust based on $\mathcal{L}_{mse}$.



(a) MAPE (b) RMSPE

Fig. 5: Performance on different α_0 in N.Y.C. Dataset.

5 Related Work

In this section, we briefly introduce related works about demand prediction and spatial-temporal computing.

5.1 Demand prediction

Demand prediction is a popular topic in urban computing, which is helpful to balance resources. Some works divided the city into grids to predict grid-level demands [15,22]. Considering about the changing of a city, some works used dynamic clusters to mine some demand patterns in a city. Chen *et al.* [3] proposed a dynamic cluster method according to correlations between stations over time. Li *et al.* [14,21] used interactions and correlations of demands to find clusters and then considered inter-cluster transition. To predict the fine-grained demands, researchers focus on station-level demands, which is more challenging. Yoon *et al.* [8] extracted a temporal pattern of stations by using similarity and then built a temporal model based on ARIMA. Hulot *et al.* [1] extracted traffic behaviors and used four different machine learning methods to help make online balancing operations based on predicting demands. Lately, several works involved graphs to characterize the relationship between stations. Chai *et al.* [9] used multi-graph to extract spatial information from different views. Lin *et al.* [4] proposed two architectures based on GCN. The first architecture linearly combined the hidden states from multiple GCN layers. The second one used LSTM to encoder temporal information over the hidden states from single GCN layers.

When existing works predicted station-level demands, they only predicted for the stations that existed all the time. But we consider the real-world scenario that there will be some new stations built at some timestamps.

5.2 Spatial-temporal computing

With the increasing number of spatial-temporal data, spatial-temporal computing is necessary to be involved to analyze urban trends. Recently, deep learning methods are widely used to fit the complex problems and big data in spatial-temporal computing. The main method is using CNN and RNN to extract spatial and temporal information[20,19]. Zhang *et al.* [18] and Lin *et al.* [17] leveraged CNN to capture the spatial information of each timestamp and fed them into three different temporal components, which were combined to predict grid-level crowd flow. Shi *et al.* [20] proposed a model named ConvLSTM, which captured spatial information by CNN and learned temporal information by LSTM. After GCN achieved a great success, it is widely used in spatial computing. Chen *et al.* [16] conducted a graph of the traffic network and used GCN to extract spatial information. To consider the multi-view of a city, Sun *et al.* [10] fused views of different periods to predict the flows in irregular regions.

Although the works above achieved great success in spatial-temporal computing, they cannot be used directly for station-level demand prediction. The reason is that they didn't consider time-evolving networks with new nodes.

6 Conclusion

To predict station-level demands, we proposed a novel model named SCEG to exploit time-evolving station graphs informed by the spatial community graph. The spatial community graph, which has coarse-grained interactions in a city, provided some possible interactions and demand trends for new stations. Meanwhile, SCEG represented fine-grained station-level interactions at each timestamp using EvolveGCN. As experimental results shown, time-evolving station graphs and the spatial community graph both contributed to demand prediction for new stations. SCEG performed better than 6 baselines on both settled stations and new stations.

References

1. Hulot, P., Aloise, D. and Jena, S.D.: Towards station-level demand prediction for effective rebalancing in bike-sharing systems. In: SIGKDD'18. pp. 378-386. ACM, London, UK(2018)
2. Borgnat, P., Abry, P., Flandrin, P., Robardet, C., Rouquier, J.B. and Fleury, E.: Shared bicycles in a city: A signal processing and data analysis perspective. Advances in Complex Systems, **14**(3), 415-438(2011)
3. Chen, L., Zhang, D., Wang, L., Yang, D., Ma, X., Li, S., Wu, Z., Pan, G., Nguyen, T.M.T. and Jakubowicz, J.: Dynamic cluster-based over-demand prediction in bike sharing systems. In: UbiComp'16. pp. 841-852. ACM, Heidelberg, German(2016)
4. Lin, L., He, Z. and Peeta, S.: Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. Transportation Research Part C: Emerging Technologies **97**, 258-276(2018)
5. Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T. and Leisersen, C.E.: Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In: AAAI'20. pp.1-8 New York, USA(2020)

6. Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M. and Li, H.: T-gcn: A temporal graph convolutional network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems, 1-11(2019)
7. Manessi, F., Rozza, A. and Manzo, M.: Dynamic graph convolutional networks. Pattern Recognition **97**, 1-18(2020)
8. Yoon, J.W., Pinelli, F. and Calabrese, F.: Cityride: a predictive bike sharing journey advisor. In: MDM'12. pp. 306-311. IEEE, Washington DC, USA(2012)
9. Chai, D., Wang, L. and Yang, Q.: Bike flow prediction with multi-graph convolutional networks. In: SIGSPATIAL'18. pp. 397-400. ACM, Seattle, USA(2018)
10. Sun, J., Zhang, J., Li, Q., Yi, X. and Zheng, Y.: Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks. In: arXiv preprint arXiv:1903.07789(2019)
11. Kingma, D.P. and Welling, M.: Auto-encoding variational bayes. In: arXiv preprint arXiv:1312.6114(2013)
12. Kipf, T.N. and Welling, M.: Semi-supervised classification with graph convolutional networks. In: arXiv preprint arXiv:1609.02907(2016)
13. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: arXiv preprint arXiv:1406.1078(2014)
14. Li, Y. and Zheng, Y., 2019. Citywide bike usage prediction in a bike-sharing system. IEEE Transactions on Knowledge and Data Engineering. **32**(6),1079-1091(2019)
15. Ye, J., Sun, L., Du, B., Fu, Y., Tong, X. and Xiong, H.: Co-Prediction of Multiple Transportation Demands Based on Deep Spatio-Temporal Neural Network. In: SIGKDD'19. pp. 305-313. ACM, Anchorage, Alaska USA(2019)
16. Chen, C., Li, K., Teo, S.G., Zou, X., Wang, K., Wang, J. and Zeng, Z,: Gated Residual Recurrent Graph Neural Networks for Traffic Prediction. In: AAAI'19. pp. 485-492. ACM, Honolulu, USA(2019)
17. Lin, Z., Feng, J., Lu, Z., Li, Y. and Jin, D.: DeepSTN+: Context-aware Spatial-Temporal Neural Network for Crowd Flow Prediction in Metropolis. In: AAAI'19. pp.1020-1027. ACM, Honolulu, USA(2019)
18. Zhang, J., Zheng, Y. and Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. In: AAAI'17. pp.1655–1661. ACM,San Francisco, USA(2017)
19. Qin, D., Yu, J., Zou, G., Yong, R., Zhao, Q. and Zhang, B.,: A novel combined prediction scheme based on CNN and LSTM for urban PM 2.5 concentration. IEEE Access **7**, 20050-20059(2019)
20. Xingjian, S.H.I., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K. and Woo, W.C.: Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: NIPS'15. pp. 802-810. Montreal, Canada(2015)
21. Li, Y., Zheng, Y., Zhang, H. and Chen, L.:Traffic prediction in a bike-sharing system. In: SIGSPATIAL'15. pp.1-10. ACM, Seattle, USA, (2015)
22. Wei, H., Wang, Y., Wo, T., Liu, Y., Xu, J.: Zest: a hybrid model on predicting passenger demand for chauffeured car service. In: CIKM'16, pp. 2203-2208. Indianapolis, USA (2016)
23. Singla, A., Marco S., Gábor B., Pratik M., Moritz M., and Andreas K.: Incentivizing users for balancing bike sharing systems. In: AAAI'15. pp.1-7. ACM, Austin Texas,USA(2015)