Deep Spatial-Temporal Sequence Modeling for Multi-Step Passenger Demand Prediction

Lei Bai[a], Lina Yao[a], Xianzhi Wang[b], Can Li[a], Xiang Zhang[c]

[a] *University of New South Wales, Sydney, NSW, Australia*
[b] *University of Technology Sydney, Sydney, NSW, Australia*
[c] *Harvard University, Boston, Massachusetts, America*

Abstract

Supply-demand imbalance poses significant challenges to transportation systems such as taxis and shared vehicles (cars and bikes) and leads to excessive delays, income loss, and energy consumption. Accurate prediction of passenger demands is an essential step towards rescheduling resources to resolve the above challenges. However, existing work cannot fully capture and leverage the complex nonlinear spatial-temporal relationships within multi-modal data. They either include excessive data from weakly-correlated regions or oversight the correlations among those similar yet geographically distant regions. Moreover, these methods mainly focus on predicting the passenger demand for one future time step, whereas predictions over longer time scales are more valuable for developing efficient vehicle deployment strategies. We propose an end-to-end deep learning based framework to solve the above challenges. Our model comprises three parts: 1) a cascade graph convolutional recurrent neural network to extract spatial-temporal correlations within city-wide historical vehicle demand data; 2) two multi-layer LSTM networks to represent the external meteorological data and time meta separately; 3) an encoder-decoder module to fuse the above two parts and decode the rep-

resentation to achieve prediction over a longer time period into the future. We evaluate our framework on three real-world datasets and show that our model can better capture the spatial-temporal relationships and outperform the most discriminative state-of-the-art methods.
Keywords: Passenger Demand Prediction, Spatial-Temporal Correlations, Graph Convolutional Network, Long-Short Term Memory.

1. Introduction

With the accelerated urbanization, around 70% of the world's population is expected to live in cities by 2050 [1]. This rapid progress has engendered significant challenges, such as increased energy consumption and traffic shortage, to transportation systems. Transforming cities to smart cities with IoT and machine learning solutions will allow for the better use of public transport and energy resources. A good example is on-demand vehicle sharing services (e.g., Didi, Uber, Mobike), which allow customers to book a shared ride through mobile apps. Such services serve massive passengers on a daily basis and have already become a critical part of the smart transportation ecosystem.

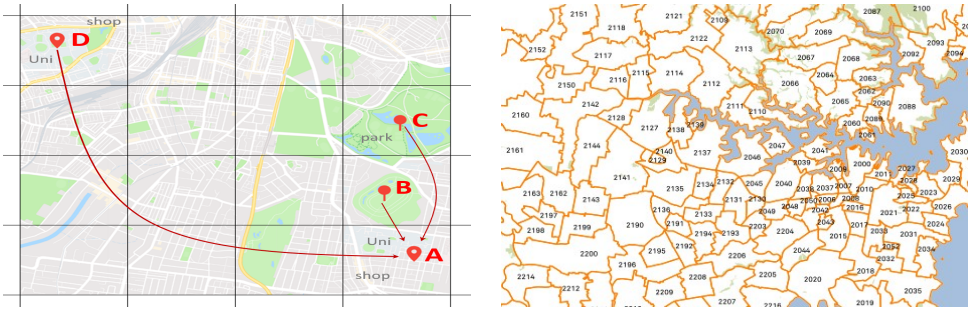
However, such services often experience a supply-demand imbalance problem. One the one hand, drivers often have to drive a long way before they can find passengers due to low demand volumes in their proximity; on the other hand, passengers may experience long delays in obtaining rides due to high demands around their locations [2, 3]. Such an imbalance leads to excessive waiting time, income loss, and energy waste [4]. Therefore, it becomes an essential step towards better resource (e.g., cars, bikes) dispatch strategies

to accurately predict passenger demands based on heterogeneous data collected on citywide deployments [5], to address the above imbalance problem. Predicting passenger demand is challenging due to the high dynamic data, complex dependencies along temporal and spatial dimensions, and sensitivity to multiple external factors (e.g., meteorological data and time meta).

Traditional methods [6] [7] employ time series models such as Auto-Regressive Integrated Moving Average (ARIMA) and its variants to predict passenger volume [2]. Such methods can only capture the temporal correlations in the target region and often incur large prediction errors. In recent years, as passenger demand datasets become abundant, researchers are starting to apply deep learning techniques for demand prediction. These studies typically use Recurrent Neural Network (RNN) and its variants (e.g., Long-Short Term Memory (LSTM) [8] and Gated Recurrent Unit (GRU) [9] networks) to capture temporal correlations, and Convolutional Neural Network (CNN) [10] to extract spatial relationships from the whole city or geographically nearest regions [11] [12] [13][14] . More recently, Convolutional LSTM (ConvLSTM) [15] is proposed to extract spatial-temporal correlations.

Despite the extensive research in passenger demand prediction, existing methods suffer from the following drawbacks:

- While spatial correlations of passenger demands have shown to boost the prediction accuracy, it is still under-explored. Recent CNN-based methods consider passenger demand in one region to be influenced by all the regions of the city [12] [13][16][14] or by its local neighbouring regions [11]. These methods either introduce irrelevant information by including excessive data from weakly correlated regions or fail to



(a) (b)

Figure 1: (a): None-Euclidean spatial correlations in the citywide passenger demand; (b): An illustration of partitioning a city into irregular regions.

capture the global spatial correlations (e.g., hidden correlations of geographically distant regions) precisely. Take the regions in Figure 1.(a) as an example, although Region A is geographically close to Region B and Region C, they have totally different points of interests. On the other hand, Region A has similar points of interests (i.e., university and shopping centers) with Region D. Thus, the passenger demand in region A shows a stronger relationship with D rather than B and C.

- Many existing studies focus on predicting passenger demands for the immediate future, i.e., next time step. However, predicting the passenger demand over a longer time scale (i.e., multi-step ahead prediction) is more valuable for transportation service operators to design better vehicle deployment strategies.

- Existing methods are specific to the way that the target city is partitioned. In particular, the majority of current approaches use an abstraction wherein the city is divided into small grids (such as 1km $\times$

1km area) [14, 17] and ignore the semantic information such as road networks and city administrative zones [2, 3]. Figure 1.(b) gives an illustration about partitioning a city into small regions by postcode, which provides convenience to extract the semantic representation about the area such as the population and income.

To address these problems, we develop an end-to-end deep learning method for multi-step citywide passenger demand forecasting based on the historical demand and the heterogeneous external data. The framework consists of three parts. Firstly, we use a cascade graph convolutional recurrent neural network module to extract the spatial-temporal correlations from the citywide historical demand. To model the spatial correlations, we treat a city as a graph where each region is a specific node. Next, an adjacency matrix of the graph is pre-defined according to the similarities between historical passenger demands of different regions. At the same time, an learnable connectivity matrix is integrated for capturing more comprehensive spatial correlations. Then, we apply Graph Convolutional Network (GCN) to the graph to extract the shared patterns only within closely related regions based on the predefined and learned graph structure jointly in a multi-graph manner [18, 19]. Our method can accurately capture spatial correlations by emphasizing regions with similar demand patterns and ignoring the noise from weakly related regions, regardless of the geographic locations. In addition, our approach does not presuppose one particular abstraction of the city, be it grid based or road network based. Secondly, we use two multi-layer LSTM networks to extract representations of the external meteorological data and time meta, respectively. Thirdly, we fuse the aforementioned components

into a joint hidden representation and decode it under an encoder-decoder structure to generate the multi-step prediction. We have evaluated our approach with three real-world passenger demand datasets of different scales: DidiSY, TaxiBJ and BikeNYC, covering different service types ranging from car sharing, regular taxis and bike sharing, respectively. The experimental results demonstrate that our method consistently outperforms a set of baselines and state-of-the-art methods. Overall, the main contributions of this work are:

- We proposed an end-to-end deep learning framework for multi-step passenger demand prediction considering the joint influence of both historical demand and heterogeneous external data.
- We propose to capture the spatial correlations in the citywide passenger demand by graph convolutional networks and infer the spatial connectivity from data directly.
- We conduct extensive experiments on three real-world datasets and demonstrate the effectiveness of our approach on both grid regions and irregular regions for both next-step demand prediction and multi-step demand prediction.

2. Related Works

Passenger demand prediction has attracted a lot of attentions due to its significance in improving citizens' quality of life. A traditional approach is considering the passenger demand as time series data [20, 21] and apply widely used time series algorithms. Li et al. [6] developed an improved

ARIMI model to forecast the citywide passenger variations in the hotspot areas. Moreira-Matias et al. [7] integrated three time-series analysis techniques (i.e., Time-Varying Poisson Model, ARIMA model, and Weighted Time-Varying Poisson Model) to make a prediction. Other traditional approaches use classical machine learning algorithms such as support vector machine (SVM) and k-nearest neighbors (KNN). Li et al. [22] proposed a short-term traffic demand prediction model with the least squares support vector machine (LS-SVM). These traditional methods neither model the nonlinear spatial-temporal correlations accurately nor do they take into account external features such as meteorological data, resulting in large predicting errors.

In recent years, deep learning techniques, which have been successfully used in various application domains, have also been used for passenger demand prediction. Liu et al. [23] designed a hybrid model using stacked auto-encoder. They first pre-train a Stacked Auto-Encoder (SAE) with all input features. Next, the pre-trained SAE is used to initialize the supervised fully connected layers to generate the prediction. Wang et al. [24] proposed a deep learning framework based on the fully-connected layers and the residual networks to forecast the gap between taxi supply and passenger demand. However, these two methods, both of which consist of fully connected layers, cannot accurately model the complex spatial-temporal relationships. Yu et al. [25] use LSTM networks to learn the temporal correlations in the traffic data and use the auto-encoders to represent static features. Lai et.al [26] combine the skip connection with LSTM to learn both long and short temporal correlations. However, none of these methods consider the spatial

correlations explicitly.

To jointly capture the spatial-temporal relationships within the citywide historical demand, Ke et al. [16] proposed to use ConvLSTM. Their method stacks multiple ConvLSTM layers and CNN layers to process historical demand data and travel time data, respectively. Similarity, Zhu et.al. [27] also deploy ConvLSTM for learning spatial-temporal correlations. They further integrate the attention mechanism to emphasize the effects of latent mobility regularities. Zhang et al. [12] come up with a spatial-temporal model to forecast the crowd flow. They organize the historical crowd flow as three Segments denoting "recent time data", "near history data", and "distant history data". Based on these fragments, they represent citywide crowd flow as a multi-dimensional image and use CNN and residual networks to extract spatial relationships [2]. One disadvantage of these two methods is that they take excessive weakly correlated regions into account, which introduces noise in capturing spatial correlations and thus decreases the prediction accuracy. To solve the above problem, Yao et al. [11] proposed "local CNN" for capturing spatial correlations only within the geographically near regions and build a weighted graph to serve as the similarity among different regions. Instead of using the data from the entire city, this method can filter weakly correlated remote regions. Their method needs to extract information about neighbouring regions and generate predictions for each region separately, which is not computationally efficient in practice. Wang et.al [14] further extend [11] with Generative Adversarial Network (GAN) to model the stochastic characteristics in the traffic demand data. However, all of these CNN-based models are limited to the grid-partitioned regions, where a city is partitioned to smaller

structured regions with grid partition method. Bai et.al. [2] further extend the scaleability of CNN-based methods to unstructured regions by organizing the demand data by region similarity and extract the spatial-temporal correlations with cascade CNN-LSTM networks.

To facilitate the process of capturing more accurate and non-Euclidean spatial correlations, graph convolutional network is introduced to the passenger demand prediction area due to its' ability in dealing with unstructured data in various applications (e.g., knowledge graph [28, 29], traffic speed forecasting [30, 31], and social recommendation [32, 33]). Geng et.al. propose ST-MGCN [18] and deploy graph convolutional network to extract the spatial correlations among different regions. They define three region graphs according to regions geographic distance, Point-of-Interest (PoI) distributions and road network connectivity, which contains more comprehensive and accurate spatial correlations. However, PoI distributions and road network connectivity data are not always available, which limits the model's generalization ability. Our work also models the passenger demand on graph. Different to ST-MGCN, we propose to infer the graph connectivity from passenger demand data itself (as depicted in Section 4.1).

3. Preliminary

3.1. Notations and the Problem Statement

Suppose a city is partitioned into N small regions, regardless of whether grid-based or road network based partitioning is employed. We represent the region sets as $\{r_1, r_2, ..., r_i, ...r_N\}$. At each time step t, a scalar $D_t(r_i)$ represents the passenger demand of region r_i in time step t. Respectively, a

vector $\boldsymbol{D_t} \in \mathbb{R}^N$ represents the passenger demand of all regions in time step t. Another vector $\boldsymbol{E_t}$ represent the external features in time step t. In this work, external features include the meteorological data (e.g. weather state, temperature, wind speed) and time meta (e.g., time of day, day of week, holidays), which are represented as $\boldsymbol{EM_t}$ and $\boldsymbol{ET_t}$, respectively.

Given the citywide historical passenger demand $\{\boldsymbol{D_0}, \boldsymbol{D_1}, ..., \boldsymbol{D_t}\}$ and external features $\{\boldsymbol{E_0}, \boldsymbol{E_1}, ..., \boldsymbol{E_t}\}$, the purpose is learning a forecasting function $\boldsymbol{\Gamma}(\cdot)$ that predicts the citywide passenger demand in the following τ time steps. Specially, instead of using all the historical passenger demand, we selectively consider the most recent q time steps historical data as input, which is a common practice in time series data analysis [3]. Thus, our work can be formulated as:

$$\begin{aligned}&(\boldsymbol{D_{t+1}}, \boldsymbol{D_{t+2}}, ..., \boldsymbol{D_{t+\tau}}) = \\ &\boldsymbol{\Gamma}(\boldsymbol{D_{t-q+1}}, \boldsymbol{D_{t-q+2}}, ..., \boldsymbol{D_t}; \boldsymbol{E_{t-q+1}}, \boldsymbol{E_{t-q+2}}, ..., \boldsymbol{E_t})\end{aligned} \tag{1}$$

3.2. Graph Convolutional Network

In this work, we use the Graph Convolution Network (GCN) defined in the spectral domain with graph Fourier Transform [34][35]. Taking the graph signal X and the corresponding adjacency matrix A as inputs, the spectral graph convolutional operation with kernel Θ is defined as follows:

$$\Theta \star \boldsymbol{D_t} = \Theta(L)\boldsymbol{D_t} = \Theta(U\Lambda U^T)\boldsymbol{D_t} = U\Theta(\Lambda)U^T\boldsymbol{D_t} \tag{2}$$

where L is the normalized graph Laplacian, $U = [u_1, u_2, ..., u_N] \in R^{N\times N}$ is the corresponding matrix of eigenvectors and $\Lambda \in R^{N\times N}$ is the diagonal matrix of eigenvalues of L [3]. According to [35], the filter Θ can be restricted

to a polynomial of Λ as:

$$\Theta(\Lambda) \approx \sum_{k=0}^{K} \theta_k \Lambda^k \tag{3}$$

where $\boldsymbol{\theta_k} \in R^K$ is a vector of Chebyshev coefficients and K is a parameter that determines the maximum radius of the convolution from central nodes [36]. Equation 3 can be further approximated in linear time when set $K = 1$ and the largest eigenvalue of L to 2:

$$\Theta \star \boldsymbol{D_t} \approx \boldsymbol{\theta_0 D_t} - \boldsymbol{\theta_1}(P^{-\frac{1}{2}} A P^{-\frac{1}{2}})\boldsymbol{D_t} \tag{4}$$

where $\boldsymbol{\theta_0}$, $\boldsymbol{\theta_1}$ are free parameters of the kernel, $P \in R^{N \times N}$ is the diagonal degree matrix of the graph with $P_{ii} = \sum_j A_{ij}$. In practice, the parameters in Equation. 4 can be further constrained to avoid overfitting and exploding/vanishing gradients by setting $\boldsymbol{\theta_0} = -\boldsymbol{\theta_1} = \boldsymbol{\theta}$. By generalizing the calculation to multiple dimensions, the first-order GCN layer is finally defined as:

$$Z^{l+1} = (I_N + P^{-\frac{1}{2}} A P^{-\frac{1}{2}}) Z^l \Theta \tag{5}$$

where $Z^l \in R^{N \times C}$ with C dimensions, $\Theta \in R^{C \times F}$ and $Z^{l+1} \in R^{N \times F}$ with F dimensions are input, learnable parameters and output of the GCN layer, respectively. This formulation can be efficiently implemented. As K is simplified to 1, successively applying k GCN layers can capture correlations from k_{th}-order neighbours of a node.

4. Model

Figure 2 illustrates the framework of our proposed method based on the encoder-decoder architecture. The encoder encodes all inputs to a joint representation, and the decoder subsequently decodes the representation into

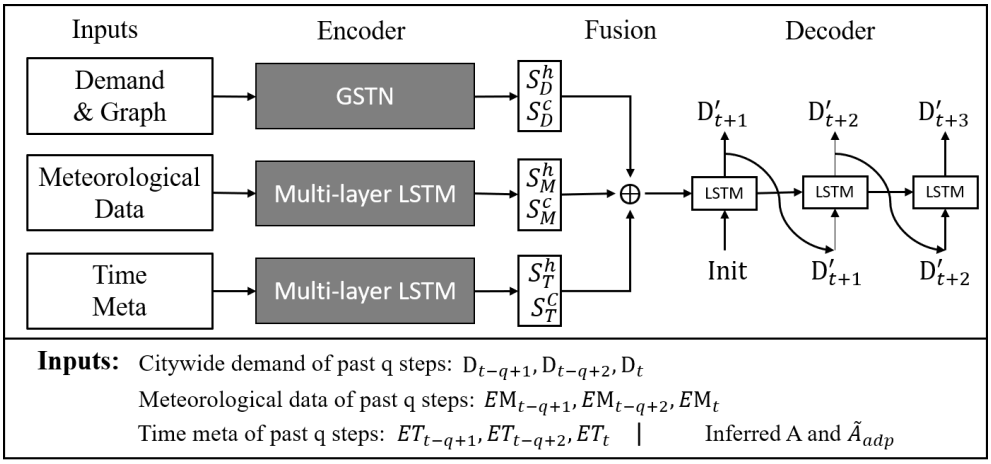


Figure 2: Proposed Framework. The framework is based on the encoder-decoder architecture. The encoder consists of a graph spatial-temporal network and two multi-layer LSTM network to generate representations for citywide historical demand, meteorological data and time meta (i.e., time of day, day of week, and holidays information), separately. The decoder is another multi-layer LSTM network decoding the fused joint representation into multi-step predictions.

a sequence of predictions. Specifically, the encoder module includes three parts: 1) a graph spatial-temporal Network that extracts from the citywide historical passenger demand. It can extract shared patterns exclusively from similar regions and avoid the negative influence of weakly related regions. 2) two multi-layer LSTM networks that learn a better representation for meteorological data and time meta, respectively. This design considers the independence between meteorological data and time meta. 3) A Hadamard fusion method that fuses the final state of the three networks above into a joint representation. In the decoder, we use another multi-layer LSTM network to decode the joint representation and to achieve multi-step prediction. We will elaborate on these modules in the following.

4.1. Learning Spatial Correlations

We first introduce how to capture the spatial correlations among different regions. Previous works assume that the passenger demand in one region is influenced by other regions. They either apply CNN to capture global spatial influences over the entire city [16] [12] [13] or local influences from geographic near regions [11]. Distinct from these existing studies, we assume that spatial correlations only depend on regions with similar demand patterns, while independent of geographic locations. Passenger demand of remote regions with similar attributes (such as PoIs, functions) could also share similar demand patterns, and vice versa. Thus, existing methods overstate the globality and proximity in passenger demand. They either introduces excessive noise from weakly related regions or neglect the correlations from remote similar regions.

Therefore, we treat the city as a graph $G = (\nu, \xi, A)$, where ν is the set of regions $\nu = \{r_i | i = 1, 2, ...N\}$ in the city, ξ is a set of edges and A is an adjacent matrix defining the spatial connectivity between different nodes (i.e, regions). Different with the works in the traffic forecasting domain where the adjacency can be directly obtained from the road network, the adjacency among different regions is unknown in passenger demand forecasting. To solve the issue, we propose to infer the connectivity of the graph from the passenger demand data in two ways. First, we pre-define the connectivity of the graph according to the historical passenger demand similarity among regions because similar regions normally have similar passenger demand.

$$A_{i,j} = \begin{cases} 1, & \text{if } Pearson(D_{0\sim t}(r_i), D_{0\sim t}(r_j)) > \beta \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where β is a threshold to control the sparsity of the pre-defined adjacent

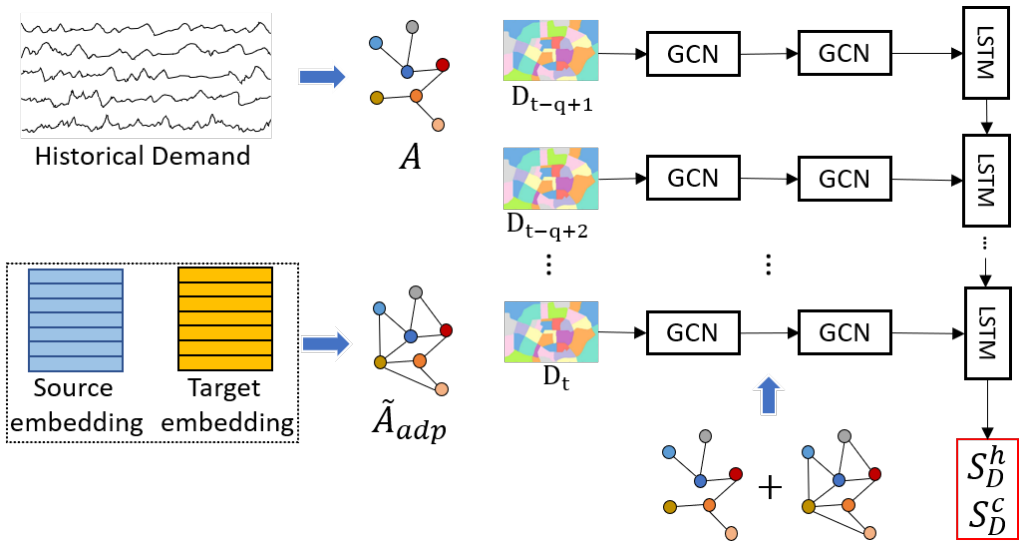


Figure 3: Graph Spatial-Temporal Network (GSTN).

matrix A, $Pearson(\cdot)$ represents the Pearson Coefficient of two data streams, $D_{0\sim t}(r_i)$ and $D_{0\sim t}(r_j)$ denote the historical passenger demand sequence of region r_i and r_j from time step 0 to t in the training data, respectively.

Second, we adaptively learn the adjacency among nodes during training based on the correlations of regions representation as proposed in [37]. Take $E_1 \in R^{N\times k}$ as the learnable source nodes embedding matrix and $E_2 \in R^{N\times k}$ as the learnable target nodes embedding matrix (where k is the embedding dimension), then:

$$\tilde{\mathbf{A}}_{adp} = \text{SoftMax}\left(\text{ReLU}\left(\mathbf{E}_1\mathbf{E}_2^T\right)\right) \tag{7}$$

Instead of using $\tilde{\mathbf{A}}_{adp}$ as the adjacent matrix of the passenger demand graph, we directly set $\tilde{\mathbf{A}}_{adp}$ to $P^{-\frac{1}{2}}AP^{-\frac{1}{2}}$, which help avoid the repetitive Laplacian transformation during training. Both E_1 and E_2 are randomly initialized and updated in the training process.

Based on the pre-defined adjacent matrix A and the learnable $\tilde{\mathbf{A}}_{adp}$, the spatial correlations among different regions can be captured by multiple stacked GCN layers in our Graph Spatial-Temporal Network (GSTN) module. As shown in Fig. 4.1, the inputs of the GSTN are citywide passenger demand for the past q time steps, the pre-defined adjacent matrix A of the city region graph, and the learnable $\tilde{\mathbf{A}}_{adp}$. At each time step, we feed the citywide passenger demand of the current time step into a set of connected GCN layers. Following Equation. 5, the calculation for each GCN layer combining A and $\tilde{\mathbf{A}}_{adp}$ is:

$$Z^{l+1} = (I_N + P^{-\frac{1}{2}} A P^{-\frac{1}{2}}) Z^l \Theta_1 + (I_N + \tilde{\mathbf{A}}_{adp}) Z^l \Theta_2 \tag{8}$$

Considering that the adjacency concludes both historical passenger demand similarity and regions representation similarity, our design can accurately capture spatial correlations from most related regions and exclude weakly correlated regions, regardless of their geographic locations.

4.2. Learning Temporal Correlations

The representations extracted from the GCN layers are then fed into a multi-layer LSTM network to capture the temporal relationships and encode citywide passenger demand of the previous q time steps into a joint representation. Notice that, we use passenger demand from the previous q time steps as input to GCN and extract the representation for each time interval separately. Correspondingly, we get q distinct representations. In Figure 4, we use a one-layer LSTM as an example to elaborate, where q outputs of GCN layers are shown as $\boldsymbol{X_{t-q+1}}, \boldsymbol{X_{t-q+2}}, ..., \boldsymbol{X_t}$. Each LSTM cell has

three inputs: $\boldsymbol{X_i}$, the cell state from last cell $\boldsymbol{C_{i-1}}$, and the output last cell $\boldsymbol{H_{i-1}}$, where $i \in [t-q+1, t]$. The calculation of each LSTM cell follows [8]:

$$\begin{aligned}
\boldsymbol{f_i} &= \sigma(\boldsymbol{W_f} \cdot [\boldsymbol{H_{i-1}}, \boldsymbol{X_i}] + \boldsymbol{b_f}) \\
\hat{\boldsymbol{f}}_{\boldsymbol{i}} &= \sigma(\boldsymbol{W_{\hat{f}}} \cdot [\boldsymbol{H_{i-1}}, \boldsymbol{X_i}] + \boldsymbol{b_{\hat{f}}}) \\
\hat{\boldsymbol{C}}_{\boldsymbol{i}} &= \sigma(\boldsymbol{W_{\hat{C}}} \cdot [\boldsymbol{H_{i-1}}, \boldsymbol{X_i}] + \boldsymbol{b_{\hat{C}}}) \\
\boldsymbol{C_i} &= \boldsymbol{f_i} * \boldsymbol{C_{i-1}} + \hat{\boldsymbol{f}}_{\boldsymbol{i}} * \hat{\boldsymbol{C}}_{\boldsymbol{i}} \\
\boldsymbol{o_i} &= \sigma(\boldsymbol{W_o} \cdot [\boldsymbol{H_{i-1}}, \boldsymbol{X_i}] + \boldsymbol{b_o}) \\
\boldsymbol{H_i} &= \boldsymbol{o_i} * tanh(\boldsymbol{C_i})
\end{aligned} \tag{9}$$

where $\boldsymbol{W_f}$, $\boldsymbol{W_{\hat{f}}}$, $\boldsymbol{W_{\hat{C}}}$, $\boldsymbol{W_o}$, $\boldsymbol{b_f}$, $\boldsymbol{b_{\hat{f}}}$, $\boldsymbol{b_{\hat{C}}}$, and $\boldsymbol{b_o}$ are parameters to be learned. The cell state $\boldsymbol{C_i}$ are transferred and updated in all LSTM cells and can be regarded as an accumulation of all previous information. So the cell state $\boldsymbol{C_t}$ generated by the last (q_{th}) LSTM cell contains all spatial-temporal information from the passenger demand of the previous q time steps. We use $\boldsymbol{C_t}$ together with $\boldsymbol{H_t}$ as the output of the LSTM network. When stacking multiple LSTM layers, the outputs are $\boldsymbol{C_t}$ and $\boldsymbol{H_t}$ of the last LSTM cell in all layers. We represent them as $(\boldsymbol{S_D^h}, \boldsymbol{S_D^c})$ in Figure 2.

4.3. Representing External Features

In addition to the spatial-temporal correlations hidden in historical city-wide demand sequence, many external factors such as time meta (e.g., time of day, day of week, holidays) and meteorological data (e.g., weather, temperature, wind speed) also influence the passenger demand [21]. For example, passenger demand tends to be very high in the morning rush hour and low at midnight. Moreover, the influence of time meta and meteorological data

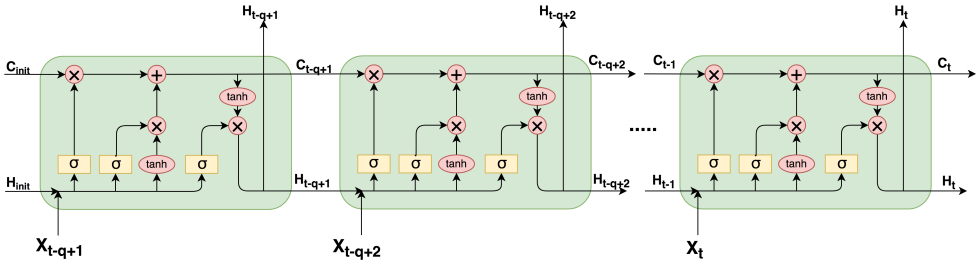


Figure 4: Illustration of one layer LSTM

on passenger demand is independent of each other [16][12]. Based on these observations, we feed the meteorological data sequence and time meta sequence of past q time steps into two separate multi-layer LSTM networks to obtain their representation. Similar to Section 4.2, we only get the state and output of the last LSTM cell in the multi-layer LSTM networks as the output and representation of inputs because they integrate all the information in the input data.

In addition, we use a fully-connected layer to map the final state of time meta part to N times the higher dimension to make it match the number of regions, based on the fact that time meta is the same among all regions. This way, we can represent meteorological data and time meta as $(\boldsymbol{S_M^h}, \boldsymbol{S_M^c})$ and $(\boldsymbol{S_T^h}, \boldsymbol{S_T^c})$.

4.4. Fusion and Prediction

In the previous parts, we introduced our design of three distinct neural networks to obtain representations of historical citywide passenger demand, meteorological data and time meta, separately. Before feeding these representations into the decoder module, they should be fused together:

$$\boldsymbol{S^h} = \boldsymbol{S_D^h} \odot W_D^h + \boldsymbol{S_T^h} \odot W_T^h + \boldsymbol{S_M^h} \odot W_M^h \tag{10}$$

$$\boldsymbol{S^c} = \boldsymbol{S^c_D} \odot W^c_D + \boldsymbol{S^c_T} \odot W^c_T + \boldsymbol{S^c_M} \odot W^c_M \tag{11}$$

where $\boldsymbol{S^h}$ and $\boldsymbol{S^c}$ are joint representation of all inputs to our model, W^h_D, W^h_T, W^h_M, W^c_D, W^c_T, W^c_M are learnable parameters, $\odot$ is element-wise hadamard product operation, which can simultaneously learn a joint representation of each region from historical passenger demand, meteorological data and time meta separately.

To predict the passenger demand in the next τ time steps, we use another LSTM networks to decode the joint representation ($\boldsymbol{S^h}$, $\boldsymbol{S^C}$). Similar to the methods in [9] and [38], the inputs of the decoder are comprised of two parts: the encoded representation of encoder, and an initial variable. The encoded representation ($\boldsymbol{S^h}$, $\boldsymbol{S^C}$) is fed into the decoder as the initial state of LSTM network (as shown in Figure 4), and the initial variable is served as the first input of LSTM network to start the decoding. In machine translation research, the most commonly used initial variable is the 'End-of-sentence" token. In our work, we use the first part of the joint representation $\boldsymbol{S^h}$ as the initial variable as it contains more information than zero variable. The inputs to the subsequent cells of the decoder are the output of the last LSTM cell, as shown in the right part of Figure 2.

4.5. Optimization and Training

The outputs of all LSTM cells in the decoder constitute the predicted passenger demand ($\boldsymbol{D'_{t+1}}, \boldsymbol{D'_{t+2}}, ..., \boldsymbol{D'_{t+\tau}}$). In the training process, our target is minimizing the error between the predicted passenger demand and the actual passenger demand. We deploy the mean squared error (MSE) of the predicted passenger demand and the true passenger demand for τ time steps

as the loss function, written as:

$$\mathcal{L}(W_\theta) = \sum_{i=1}^{i=\tau} \|\boldsymbol{D_{t+i}} - \boldsymbol{D'_{t+i}}\|_2^2 \tag{12}$$

where W_θ denotes all the learnable parameters in our network. These parameters can be obtained through the back-propagation algorithm and Adam optimizer [39]. Furthermore, we use the teacher forcing strategy in the decoder to achieve efficiency in training [3]. Specifically, we directly use the true value $\boldsymbol{D_{t+i}}$ when training the model, instead of feeding predicted $\boldsymbol{D'_{t+i}}(i \in [1, \tau - 1])$ to $i + 1_{th}$ LSTM cells in the decoder.

5. Experiments

In this part, we first introduce the datasets used for evaluating our model. Then, the experimental settings and evaluation metrics are outlined. Finally, we evaluate our methods from three perspectives, focusing on the three drawbacks in previous works (see Section 1), which demonstrates that our model can effectively solve these problems.

5.1. Datasets

We use three real-world datasets in our experiments, as detailed below:

- DidiSY: A self-collected dataset which consists of three parts: 1) the ride-sharing demand data from Didi, which is the biggest online ride-sharing company in China; 2) Time meta, which includes the time of day, day of week and holidays; 3) Meteorological data, including weather, temperature, wind speed, and visibility. The dataset is collected in Shenyang, which is a large city of China, from 05/Dec/2016

to 04/Feb/2017. Each time slot is one hour. We deploy the last six days demand as testing data and previous data as training data.

- BikeNYC [12]: The public BikeNYC dataset only consists of two parts: the bike demand part and the time meta part. This dataset covers the shared bike hire and returns data of CityBike in New York from 1 Apr 2014 to 30th Sept 2014. Each time step is one hour. To be consistent with previous work using this dataset. [12][13], the last ten days data are chosen as testing data.

- TaxiBJ [12]: The public TaxiBJ dataset contains taxi demand in Beijing from 1 Mar 2015 to 30th Jun 2015. Similar to the DidiSY dataset, TaxiBJ contains passenger demand, time meta, and meteorological data. Each time step is 30 minutes in this dataset. Last ten days data are chosen as testing data.

5.2. Experiment Settings

Before feeding the dataset into the model, we transform the categorical features (e.g., hour of day, day of week, holiday, and weather) with one-hot encoding. Besides, the continuous features including visibility, wind speed, and temperature, are scaled by the Min-Max normalization method. The passenger demand data is also normalized by the Min-Max normalization and reversed back to the actual value for evaluating the prediction accuracy. We implemented our model in Python with Pytorch 1.1. After parameter-tuning, the final model of our method employs two GCN layers followed by two LSTM layers for encoding the historical passenger demand data and three LSTM layers for representing the historical external data (i.e., time

meta and metrological data). The order of GCN is set 2 and the historical window size q is set to 8 following DMVST [11]. In the decoder, we only use one LSTM layer to generate the prediction. The batch size is set to 64 for all datasets, and the learning rate is set to 0.0001, 0.0007, and 0.001 for DidiSY, BikeNYC, and TaxiBJ, respectively. For the comparison methods, we tuned them carefully and chose the best parameters for them.

The batch size is set to 64. Historical windows size q is set to 8. For the multi-layer LSTM netowrks, we use a two-layer LSTM network for Graph Spatial-temporal correlations extraction and a three-layer LSTM network for external features representation.

At each prediction step, we deploy three evaluation metrics to evaluate the performance of our model, i.e., Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and the Mean Absolute Percentage Error (MAPE).

$$RMSE = \sqrt{\frac{1}{\epsilon}\sum_{i}(\hat{D}_{t+1}(r_i) - D_{t+1}(r_i))^2}$$

$$MAE = \frac{1}{\epsilon}\sum_{i}\|\hat{D}_{t+1}(r_i) - D_{t+1}(r_i)\|$$

$$MAPE = \frac{1}{\epsilon}\sum_{i}\|\frac{\hat{D}_{t+1}(r_i) - D_{t+1}(r_i)}{D_{t+1}(r_i)}\|$$

where ϵ is the total number of predicted values.

5.3. Evaluation on next-step prediction

We first compare our method with two representative traditional baselines and eight discriminative state-of-the-arts methods to evaluate the ability of our model in capturing the spatial-temporal correlations. Considering the

fact that most state-of-the-art methods can only achieve next-step prediction, we use our prediction in the first time step as our result in this part. The comparison methods we used are:

- HA: The historical average model forecasts the future passenger demand by calculating the averaged value of previous passenger demand in the corresponding time interval of the same region. For example, prediction of 10:00am-11:00am on Friday is made by averaging all demand from 10:00am-11:00am on historical Friday.

- OLSR: The Ordinary Least Square Regression model is a type of linear regression model, which can estimate the correlations between multiple variables.

- XGBoost [40]: XGBoost is a boosting tree-based machine learning method, which has been shown to achieve good accuracy for a wide range of applications.

- DeepST [13]: DeepST is a CNN based method which organizes historical data to three components (temporal "closeness", "period", "trend") and uses CNN to capture spatial-temporal dependencies.

- ResST-Net [12]: ResST-Net is an extension to DeepST, which further integrates residual networks to stack more CNN layers. This model captures spatial correlations from the entire city.

- DMVST-Net [11]: DMVST-Net is composed of three views: spatial view, temporal view, and semantic view. It can efficiently capture both the spatial and temporal dependencies.

- ConvLSTM [15]: ConvLSTM is deep learning method specially designed for spatial-temporal analysis. It extends LSTM by using convolutional structures in both the input-to-state and state-to-state transitions. ConvLSTM achieves multi-step prediction by integrating with encoder-decoder framework.

- FCL-Net [16]: FCL-Net combines several ConvLSTM layers and CNN to extract spatial-temporal relationships from past passenger demand sequence and average travel time.

- FlowFlexDP [41]: FlowFlexDP is a deep learning method that extracts spatial correlations from historical passenger demand and crowd outflow, it also uses GCN.

- DCRNN [42]: Similar with ConvLSTM, DCRNN extends LSTM by using diffusion convolutional operation for calculating the reset gate the update gate. This model considers multi-step prediction with integrating to encoder-decoder framework.

As can be observed from Table 1, our method consistently achieves the best performance with all three datasets, which demonstrates the superiority of our model in accurately capturing the citywide spatial-temporal correlations. More specifically, performance gains of our model over the baseline Historical Average are: 21.81% (RMSE), 20.74% (MAE) and 26.99% (MAPE) relative improvement in DidiSY dataset, 45.13% (RMSE), 40.05% (MAE) and 52.63% (MAPE) relative improvement in BikeNYC dataset, 56.47% (RMSE), 49.28% (MAE) and 48.13% (MAPE) relative improvement in TaxiBJ dataset. When comparing to the best state-of-the-art methods,

Table 1: The experimental results of next-step prediction over three datasets (the best performance is displayed in bold).

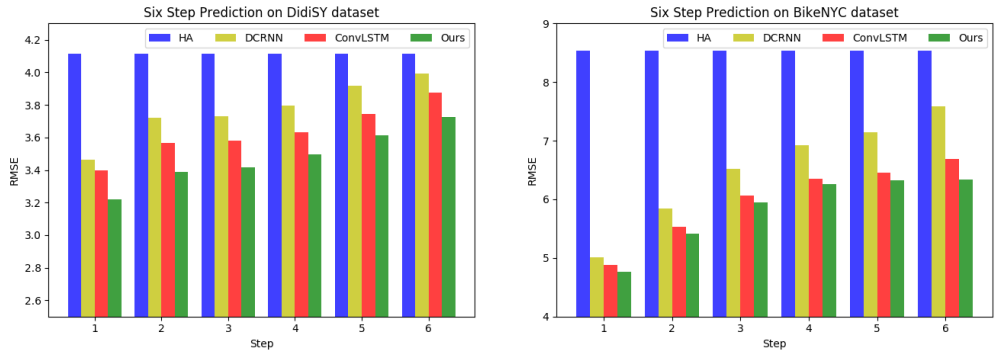
Index	Method	DidiSY			BikeNYC			TaxiBJ		
		RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
1	HA	4.112	2.646	0.426	8.541	3.695	0.437	40.439	20.696	0.268
2	OLSR	3.713	2.528	0.379	8.502	4.652	0.391	23.921	14.937	0.276
3	XGBoost [40]	3.612	2.394	0.402	6.914	3.423	0.367	22.927	13.687	0.212
4	DeepST [13]	3.362	2.221	0.337	6.603	2.549	0.242	18.305	11.264	0.157
5	ResST-Net [12]	3.449	2.331	0.318	6.159	2.432	0.228	17.649	10.599	0.141
6	DMVST-Net [11]	3.440	2.232	0.373	4.766	2.318	0.224	18.206	11.085	0.153
7	ConvLSTM [15]	3.399	2.278	0.379	4.881	2.561	0.230	19.487	12.006	0.171
8	FCL-Net [16]	3.364	2.172	0.381	4.959	2.362	0.275	18.176	10.756	0.169
9	FlowFLexDP [41]	3.292	2.143	0.336	6.003	2.801	0.271	19.538	11.945	0.160
10	DCRNN [42]	3.465	2.284	0.371	5.016	2.661	0.232	20.495	12.521	0.178
11	**Ours**	**3.219**	**2.098**	**0.311**	**4.697**	**2.223**	**0.214**	**17.601**	**10.492**	**0.139**

our model still achieves 2.21% (RMSE), 2.10% (MAE) and 2.20% (MAPE) relative improvement in DidiSY dataset, 1.44% (RMSE), 4.09% (MAE) and 4.46% (MAPE) relative improvement in BikeNYC dataset, 0.27% (RMSE), 1.01% (MAE) and 1.42% (MAPE) relative improvement in TaxiBJ dataset.

Besides, we can also observe that the RMSE and MAE is the largest in TaxiBj dataset for all methods. This is because the passenger demand is extremely high in this dataset. Although these three datasets are of totally different scale, our method can adapt well to all of them. At the same time, some methods such as ResST-Net can not achieve good prediction with the small dataset while FlowFlexDP performs poorly with the large dataset.

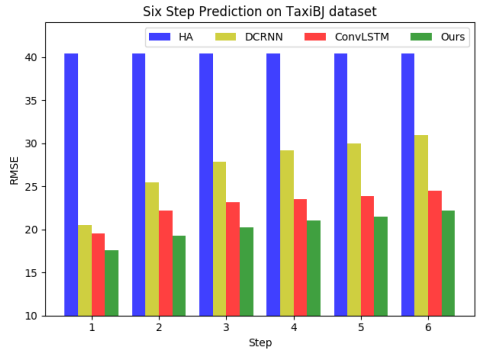
5.4. Evaluation on multi-step prediction

Next, we evaluate the ability of our model in conducting long-term forecasting, e.g., multi-step prediction. We predict the passenger demand of next

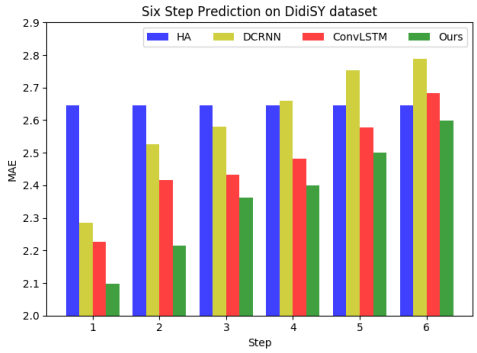


(a)

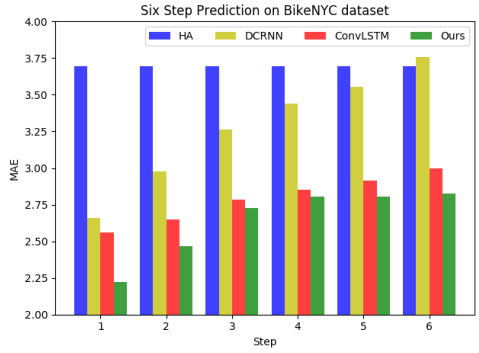
(b)



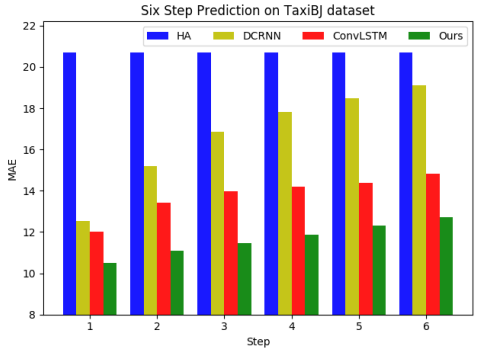
(c)



(d)



(e)



(f)

Figure 5: Evaluation on multi-step prediction. (a) - (c) RMSE of six steps prediction on the DidiSY, BikeNYC and TaxiBJ dataset, (d) - (e) MAE of six steps prediction on the DidiSY, BikeNYC and TaxiBJ dataset

six time steps and compare it to three methods: HA, DCRNN and ConvLSTM. RMSE and MAE are used as comparison metrics. As can be observed from Figure 5, the prediction results of HA remains the same for all time steps, this is reasonable as HA relies on previous days corresponding passenger demands, which are readily available for all six steps. At the same time, the prediction of other three methods deteriorate with bigger time steps, the reason is that their predictions depend on the prediction of last step(s), which would accumulate the prediction errors. Besides, DCRNN, ConvLSTM, and our model achieve the biggest relative improvement on the TaxiBJ dataset and the least relative improvement on the DidiSY dataset over HA, which demonstrates that the ability of deep learning models in modeling spatial-temporal correlations is more superior on more complex and larger dataset. Moreover, the overall performance of DCRNN and ConvLSTM is better than HA, although their long-term prediction performance (i.e., at the sixth step) would be worse than HA (e.g., on the DidiSY dataset as shown in Figure 5.(d)). In summary, it can be observed that our model performs the best consistently, and the trends of prediction degradation over three datasets are small, with a certain degree of stability compared with state-of-the-art methods.

5.5. Evaluation on prediction with irregular regions

We also evaluate the performance of our model under the situation that a city is partitioned into irregular regions with the road-network based partitioning method. The DidiSY dataset is use to conduct this experiment because it retains the geographic location (GPS) of each service request. We re-partition the city (Shengyang) to sub-regions based on the road network

Table 2: Prediction results on irregular regions

Index	Method	RMSE	MAE
1	HA	4.231	2.714
2	ARIMA	4.001	2.681
3	SARIMA	3.937	2.619
4	OLSR	3.719	2.496
5	MLP	3.699	2.436
6	XGBoost	3.533	2.341
7	FlowFLexDP	3.518	2.322
8	DCRNN	3.526	2.332
9	Ours	3.294	2.117

and re-organize our demand data to corresponding regions according to the GPS information.

Our approach is flexible and can be applied to the new dataset without modification. On the other hand, as described in Section 1 and shown in Section 5.3, most existing state-of-the-art methods use CNN to capture spatial correlations and thus cannot be applied in such circumstance. To better evaluate our method, we further add three new comparison methods: Auto-Regressive Integrated Moving Average model (ARIMA), Seasonal Auto-Regressive Integrated Moving model (SARIMA) and Multiple Layer Perceptron (MLP). The experimental results of these methods are listed in Table 2. We can observe that prediction results of this re-organized dataset are slightly higher than the results in Sec 5.3. This is because the city is partitioned into fewer regions. Besides, our model still outperforms all other

methods in these irrgular regions. The experimental results demonstrate that our approach is not constrained by the region partition methods and can achieve superior performance consistently.

6. Conclusions

Accurate prediction of future passenger demand is important for smart transportation systems. In this work, we develop a novel deep learning approach for passenger demand forecasting with historical passenger demand and external data. Our method can capture both spatial and temporal relationships among the citywide historical demand data and integrate the influence of external data into prediction effectively. By formulating the demand on graph and inferring inter-connectivity from data, our model can capture more accurate spatial relationships among regions than existing work and is irrespective to the region partition methods. Moreover, our method can produce predictions for multiple steps in one execution based on the encoder-decoder framework. Experimental results on three real-world datasets show that our approach outperforms a set of baselines and state-of-the-arts.

While our method achieves promising results by explicitly modeling the influence of both spatial dependencies, temporal correlations and external factors, one potential limitation is the infeasibility of capturing the spatio-temporal dynamics in the passenger demand data. For example, the spatial correlations among different regions may be different during the weekdays and weekends. The graph infered in our method is static and thus cannot model the dynamics accurately. In the future, we will further take spatio-temporal dynamics into consideration and capture more comprehen-

sive spatio-temporal correlations.

[1] A. K. Debnath, H. C. Chin, M. M. Haque, B. Yuen, A methodological framework for benchmarking smart transport cities, Cities 37 (2014) 47–56.

[2] L. Bai, L. Yao, S. S. Kanhere, Z. Yang, J. Chu, X. Wang, Passenger demand forecasting with multi-task convolutional recurrent neural networks, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2019, pp. 29–42.

[3] L. Bai, L. Yao, S. Kanhere, X. Wang, Q. Sheng, et al., Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting, arXiv preprint arXiv:1905.10069 (2019).

[4] R. Jia, P. Jiang, L. Liu, L. Cui, Y. Shi, Data driven congestion trends prediction of urban transportation, IEEE Internet of Things Journal 5 (2) (2018) 581–591.

[5] P. M. Santos, J. G. Rodrigues, S. B. Cruz, T. Lourenço, P. M. d'Orey, Y. Luis, C. Rocha, S. Sousa, S. Crisóstomo, C. Queirós, et al., Portolivinglab: An iot-based sensing platform for smart cities, IEEE Internet of Things Journal 5 (2) (2018) 523–532.

[6] X. Li, G. Pan, Z. Wu, G. Qi, S. Li, D. Zhang, W. Zhang, Z. Wang, Prediction of urban human mobility using large-scale taxi traces and its applications, Frontiers of Computer Science 6 (1) (2012) 111–121.

[7] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, L. Damas, Predicting taxi–passenger demand using streaming data, IEEE Transactions on Intelligent Transportation Systems 14 (3) (2013) 1393–1402.

[8] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[10] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.

[11] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, Z. Li, Deep multi-view spatial-temporal network for taxi demand prediction, in: 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, AAAI press, 2018, pp. 2588–2595.

[12] J. Zhang, Y. Zheng, D. Qi, Deep spatio-temporal residual networks for citywide crowd flows prediction, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI press, 2017, pp. 1655–1661.

[13] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, Dnn-based prediction model for spatio-temporal data, in: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2016, pp. 1–4.

[14] S. Wang, J. Cao, H. Chen, H. Peng, Z. Huang, Seqst-gan: Seq2seq generative adversarial nets for multi-step urban crowd flow prediction, ACM Transactions on Spatial Algorithms and Systems (TSAS) 6 (4) (2020) 1–24.

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

[16] J. Ke, H. Zheng, H. Yang, X. M. Chen, Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach, Transportation Research Part C: Emerging Technologies 85 (2017) 591–608.

[17] H. Yao, X. Tang, H. Wei, G. Zheng, Z. Li, Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5668–5675.

[18] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, Y. Liu, Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3656–3663.

[19] G. Wan, B. Du, S. Pan, J. Wu, Adaptive knowledge subgraph ensemble for robust and trustworthy knowledge graph completion, World Wide Web 23 (1) (2020) 471–490.

[20] H. Wang, Q. Zhang, J. Wu, S. Pan, Y. Chen, Time series feature learning with labeled and unlabeled data, Pattern Recognition 89 (2019) 55–66.

[21] Y. Liang, S. Ke, J. Zhang, X. Yi, Y. Zheng, Geoman: multi-level attention networks for geo-sensory time series prediction, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 3428–3434.

[22] Y. Li, J. Lu, L. Zhang, Y. Zhao, Taxi booking mobile app order demand prediction based on short-term traffic forecasting, Transportation Research Record 2634 (1) (2017) 57–68.

[23] L. Liu, R.-C. Chen, A novel passenger flow prediction model using deep learning methods, Transportation Research Part C: Emerging Technologies 84 (2017) 74–91.

[24] D. Wang, W. Cao, J. Li, J. Ye, Deepsd: Supply-demand prediction for online car-hailing services using deep neural networks, in: 2017 IEEE 33rd international conference on data engineering (ICDE), IEEE, 2017, pp. 243–254.

[25] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, Y. Liu, Deep learning: A generic approach for extreme condition traffic forecasting, in: Proceedings of the 2017 SIAM international Conference on Data Mining, SIAM, 2017, pp. 777–785.

[26] G. Lai, W.-C. Chang, Y. Yang, H. Liu, Modeling long-and short-term temporal patterns with deep neural networks, in: The 41st International

ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 95–104.

[27] X. Zhou, Y. Shen, Y. Zhu, L. Huang, Predicting multi-step citywide passenger demands using attention-based neural networks, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 736–744.

[28] S. Ji, S. Pan, E. Cambria, P. Marttinen, P. S. Yu, A survey on knowledge graphs: Representation, acquisition and applications, arXiv preprint arXiv:2002.00388 (2020).

[29] T. Guo, S. Pan, X. Zhu, C. Zhang, Cfond: consensus factorization for co-clustering networked data, IEEE Transactions on Knowledge and Data Engineering 31 (4) (2018) 706–719.

[30] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, J. Zhang, Urban traffic prediction from spatio-temporal data using deep meta learning, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1720–1730.

[31] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, C. Zhang, Connecting the dots: Multivariate time series forecasting with graph neural networks, arXiv preprint arXiv:2005.11650 (2020).

[32] F. Xiong, W. Shen, H. Chen, S. Pan, X. Wang, Z. Yan, Exploiting implicit influence from information propagation for social recommendation, IEEE Transactions on Cybernetics (2019).

[33] F. Xiong, X. Wang, S. Pan, H. Yang, H. Wang, C. Zhang, Social recommendation with evolutionary opinion dynamics, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2018).

[34] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, arXiv preprint arXiv:1312.6203 (2013).

[35] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

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

[37] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 1907–1913.

[38] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[39] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[40] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[41] J. Chu, K. Qian, X. Wang, L. Yao, F. Xiao, J. Li, X. Miao, Z. Yang, Passenger demand prediction with cellular footprints, in: 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), IEEE, 2018, pp. 1–9.

[42] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, arXiv preprint arXiv:1707.01926 (2017).