Spatio-Temporal Graph Convolutional and Recurrent Networks for Citywide Passenger Demand Prediction

ABSTRACT

Online ride-sharing platforms have become a critical part of the urban transportation system. Accurately recommending hotspots to drivers in such platforms is essential to help drivers find passengers and improve users' experience, which calls for efficient passenger demand prediction strategy. However, predicting multi-step passenger demand is challenging due to its high dynamicity, complex dependencies along spatial and temporal dimensions, and sensitivity to external factors (meteorological data and time meta). We propose an end-to-end deep learning framework to address the above problems. Our model comprises three components in pipeline: 1) a cascade graph convolutional recurrent neural network to accurately extract the spatial-temporal correlations within citywide historical passenger demand data; 2) two multi-layer LSTM networks to represent the external meteorological data and time meta, respectively; 3) an encoder-decoder module to fuse the above two parts and decode the representation to predict over multi-steps into the future. The experimental results on three real-world datasets demonstrate that our model can achieve accurate prediction and outperform the most discriminative state-of-the-art methods.

KEYWORDS

Passenger Demand Prediction; Spatial-Temporal Correlations; Deep Learning;

1 INTRODUCTION

Online ride-sharing platforms such as Didi and Uber have become a critical part of the urban transportation system and serves a large number of passengers on a daily basis. Despite the adoption of sophisticated ICT technologies, drivers in such platforms still lack knowledge about the locations of potential passengers. As a result, they often have to drive a long way before finding a passenger due to low demand volumes in their proximity. This issue not only incurs income loss and energy waste to drivers but also leads to excessive waiting time to users and harms users experience. Thus, recommending regions with high probability of finding passengers, namely hotspots, for drivers in a timely manner is pressing for online ride-sharing platforms [8]. In general, citywide passenger demand prediction which aims at forecasting the future (next one or few time steps) variations of passenger demand in each region is the fundamental step for recommending hotspots [3].

However, predicting multi-step passenger demand still remains challenging: 1) passenger demand in a region is influenced by other regions' demand in the city. Accurately capturing these spatial correlations requires to not only find out closely related regions but also filter out weakly related regions to avoid irrelevant interference; 2) as the typical time series data, passenger demand changes over time and fluctuates tremendously; 3) the passenger demand is sensitive to external factors such as meteorological data (e.g. rain) and time meta (e.g. morning rush hour).

To tackle these challenges, we propose an end-to-end trainable framework to achieve efficient multi-step passenger demand prediction. The framework consists of three components. Firstly, we design 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. The adjacency matrix of the graph is generated according to the similarities between historical passenger demands of different regions. Then, we apply Graph Convolutional Network (GCN) to the graph to extract the shared patterns only within closely related regions. Our method can precisely learn spatial correlations by emphasizing regions with similar demand patterns and ignoring the noise from weakly related regions, regardless of the geographcial proximity. 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. The experimental results demonstrate that our method consistently outperforms a set of baselines and state-of-the-art methods.

2 NOTATIONS AND PROBLEM STATEMENT

Suppose a city is partitioned into N small regions, irrespective of whether grid 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. The vector $\boldsymbol{E_t}$ represents 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 $\{\boldsymbol{E_0}, \boldsymbol{E_1}, ..., \boldsymbol{E_t}\}$, our target is to learn a prediction function $\boldsymbol{\Gamma}(\cdot)$ that forecasts the citywide passenger demand in the next τ ($\tau > 1$) time steps:

$$(\boldsymbol{D_{t+1}}, \boldsymbol{D_{t+2}}, ..., \boldsymbol{D_{t+\tau}}) = \boldsymbol{\Gamma}(\boldsymbol{D_0}, \boldsymbol{D_1}, ..., \boldsymbol{D_t}, \boldsymbol{E_0}, \boldsymbol{E_1}, ..., \boldsymbol{E_t}) \quad (1)$$

3 MODEL

Figure 1 illustrates the framework of our proposed method based on the encoder-decoder architecture. The encoder encodes all inputs into a joint representation, and the decoder subsequently decodes the representation into a sequence of predictions. Specifically, the encoder module includes three parts: 1) a graph spatial-temporal

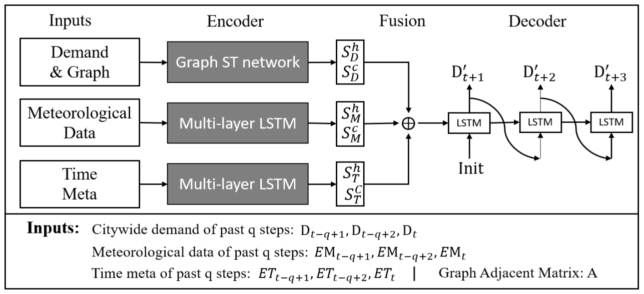

Inputs
Encoder
Fusion
Decoder
Demand & Graph
Graph ST network
S_D^h S_D^c
Meteorological Data
Multi-layer LSTM
S_M^h S_M^c
Time Meta
Multi-layer LSTM
S_T^h S_T^C
D'_{t+1}
D'_{t+2}
D'_{t+3}
LSTM
LSTM
LSTM
Init
Inputs: Citywide demand of past q steps: $D_{t-q+1}, D_{t-q+2}, D_t$
Meteorological data of past q steps: $EM_{t-q+1}, EM_{t-q+2}, EM_t$
Time meta of past q steps: $ET_{t-q+1}, ET_{t-q+2}, ET_t$ | Graph Adjacent Matrix: A

Figure 1: Proposed Framework

network extracts correlations from the citywide historical passenger demand. It extracts the shared patterns exclusively from similar regions and avoids 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 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.

3.1 Graph Spatial-Temporal Network

We first introduce the design of graph spatial-temporal network to process the historical citywide passenger demand data. For extracting the spatial correlations, 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 [6] [7] or local influences from geographic near regions [5]. 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 introduce excessive noise from weakly related regions or neglects the correlations from remote similar regions.

In this regard, we treat the city as a graph $G = (v, \xi, A)$, where v is the set of regions $v = \{r_i | i = 1, 2, ...N\}$, ξ is a set of edges and A is an adjacent matrix. We define the connectivity of the graph according to the passenger demand pattern similarity among regions.

$$A_{i,j} = \begin{cases} 1, & \text{if } Similarity_{r_i,r_j} > \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where ϵ is a threshold to control the sparsity of matrix A.
In order to find the regions with similar patterns, a direct way is to calculate the correlations (e.g., Pearson Coefficient) by historical demand. Let $D_{0\sim t}(r_i)$ represent historical order sequence of region r_i from time 0 to t in the training data. Then the similarity of region r_i and r_j can be defined as:

$$Similarity_{r_i,r_j} = Pearson(D_{0\sim t}(r_i), D_{0\sim t}(r_j)) \tag{3}$$

As shown in Figure 2, the inputs of the graph spatial-temporal module are citywide passenger demand for the past q time steps and the calculated adjacency matrix A of the city region graph.

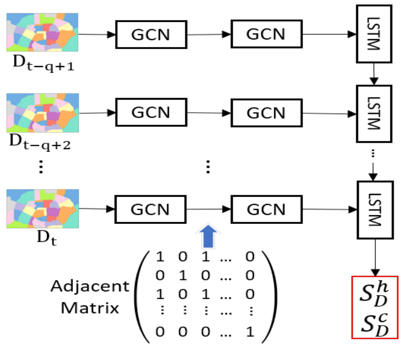

D_{t-q+1}
D_{t-q+2}
D_t
GCN
GCN
LSTM
Adjacent Matrix
$\begin{pmatrix} 1 & 0 & 1 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}$
S_D^h S_D^c

Figure 2: Graph Spatial-temporal Network.

At each time step, we feed the citywide passenger demand of the current time step into a set of connected GCN layers.
GCN Layer In this work, we use the graph convolution operation defined in the spectral domain with graph Fourier Transform [1]. Taking the graph signal $\boldsymbol{D_t}$ at time step t and adjacency matrix A as inputs, the spectral graph convolutional operation with kernel Θ is defined as follows:

$$\Theta \star \boldsymbol{D_t} = \Theta(L)\boldsymbol{D_t} = \Theta(U\Lambda U^T)\boldsymbol{D_t} = U\Theta(\Lambda)U^T\boldsymbol{D_t} \tag{4}$$

where Θ is the graph convolution operator, $U \in \mathbb{R}^{N\times N}$ is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - P^{-\frac{1}{2}}AP^{-\frac{1}{2}} = U\Lambda U^T \in \mathbb{R}^{N\times N}$, where I_N is the identity matrix, A is the adjecency matrix of the graph and $P \in \mathbb{R}^{N\times N}$ is the diagonal degree matrix with $P_{ii} = \sum_j A_{ij}$. However, calculating graph convolution on Eq.(4) is computationally expensive. To overcome this problem, Chebyshev polynomials are used to expand and approximate $\Theta(\Lambda)$. Kipf et.al [1] further proposed to set the order of Chebyshev polynominals to 1 and approximate the largest eigenvalue of L to 2. Finally, the approximated efficient calculation of graph convolution layer is generalized as follows:

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

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. 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. Considering that our graph is generated by passenger demand similarity, our design can automatically capture spatial correlations from most related regions while excluding weakly correlated regions, regardless of their geographic proximity.
LSTM Layer 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 3, 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 cell state $\boldsymbol{C_i}$ is 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

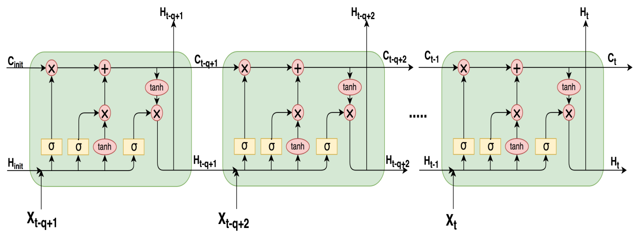


Figure 3: Illustration of one layer LSTM

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 1.

3.2 Representation of External Features

In addition to the spatial-temporal correlations hidden in historical citywide demand, 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. 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 on passenger demand is independent of each other [6]. Based on these observations, we feed the meteorological data and time meta of past q time steps into two separate multi-layer LSTM networks to obtain their representation. Similar to Section 4.1, 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 $(\boldsymbol{S_T^h}, \boldsymbol{S_T^c})$.

3.3 Fusion and Decoding

In Sections 3.1 and 3.2, 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:

$$\boldsymbol{S^h} = \boldsymbol{S_D^h} \odot W_D^h + \boldsymbol{S_T^h} \odot W_T^h + \boldsymbol{S_M^h} \odot W_M^h \tag{6}$$

$$\boldsymbol{S^c} = \boldsymbol{S_D^c} \odot W_D^c + \boldsymbol{S_T^c} \odot W_T^c + \boldsymbol{S_M^c} \odot W_M^c \tag{7}$$

where $\boldsymbol{S^h}$ and $\boldsymbol{S^c}$ are joint representations of all inputs to our model, $W_D^h, W_T^h, W_M^h, W_D^c, W_T^c, W_M^c$ 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 $(\boldsymbol{S^h}, \boldsymbol{S^C})$. 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 3), 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 the 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 1.

3.4 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 objective is to minimize the error between predicted passenger demand and true passenger demand. We define the loss function as the mean squared error of the predicted passenger demand and the actual passenger demand for τ time steps, written as:

$$\mathcal{L}(W_\theta) = \sum_{i=1}^{i=\tau} \|\boldsymbol{D_{t+i}} - \boldsymbol{D'_{t+i}}\|_2^2 \tag{8}$$

where W_θ represents all the learnable parameters in the network.

4 EXPERIMENTS

4.1 Experiment Settings

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

- DidiSY: This is a self-collected dataset that consists of three parts: 1) share car demand data from Didi, the biggest online ridesharing company in China; 2) Time meta, including time of day, day of week and holidays; 3) Meteorological data, including weather, temperature, wind speed, and visibility. This dataset is collected from 5/Dec/2016 to 4/Feb/2017 in Shenyang, a large city in China. Each time step is one hour. We use the last 6 days demand as test data.
- BikeNYC [6]: The public BikeNYC dataset 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 30/Sep/2014. Each time step is one hour. To be consistent with previous work using this dataset. [6][7], the last ten days data are chosen as testing data.
- TaxiBJ [6]: The public TaxiBJ dataset contains taxi demand in Beijing from 1/Mar/2015 to 30/Jun/2015. Similar to the DidiSY dataset, TaxiBJ contains passenger demand, time meta, and meteorological data. Each time step is 30 minutes and last ten days data are chosen as testing data.

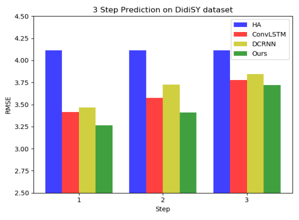
During training, we use Adam optimizer as the optimization function. After parameter tuning, we set historical demand length q to 12, batch_size to 32, learning rate to {0.0001, 0.0007, 0.001} for {DidiSY, BikeNYC, TaxiBJ}. Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) are used as evaluation metrics to evaluate the model.

4.2 Evaluation on next-step prediction

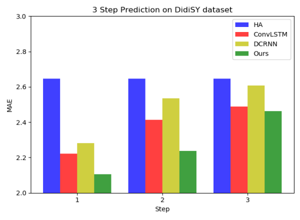
We first compare our method with with two representative traditional baselines: Historical Average (HA), Ordinary Linear Regression (OLR), and five discriminative state-of-the-art methods including DeepST [7], ResST-Net [6], DMVST-Net [5], ConvLSTM [4] and DCRNN [2] to evaluate the ability of our model in capturing

Table 1: Evaluation on next-step prediction over three datasets of different scales (best performance displayed in bold).

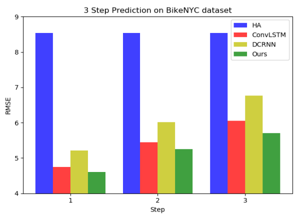
Index	Method	DidiSY			BikeNYC			TaxiBJ		
		RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
1	HA	4.112	2.646	0.426	8.541	3.695	0.437	40.439	20.696	0.268
2	OLSR	3.713	2.528	0.379	8.502	4.652	0.391	23.921	14.937	0.276
3	DeepST [7]	3.362	2.221	0.337	6.603	2.549	0.242	18.305	11.264	0.157
4	ResST-Net [6]	3.449	2.331	**0.319**	6.159	2.432	0.228	17.649	10.599	0.141
5	DMVST-Net [5]	3.440	2.232	0.373	4.766	2.318	0.224	18.206	11.085	0.153
6	ConvLSTM [4]	3.414	2.222	0.379	4.745	2.435	0.226	18.788	11.461	0.163
7	DCRNN [2]	3.465	2.281	0.371	5.215	2.776	0.241	20.569	12.517	0.177
8	**Ours**	**3.263**	**2.105**	0.323	**4.605**	**2.275**	**0.211**	**17.598**	**10.473**	**0.138**



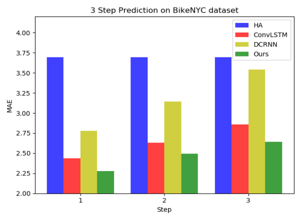
(a)



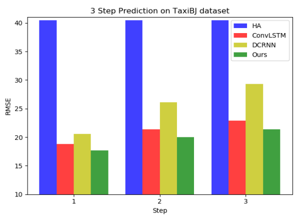
(b)



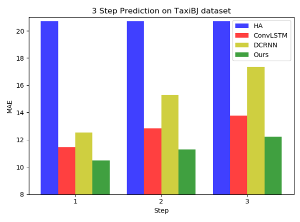
(c)



(d)



(e)



(f)

Figure 4: Evaluation on multi-step prediction.

the spatial-temporal correlations. All methods have the same input data source with our model except for HA, which only utilizes the passenger demand data. 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.

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: 20.66% (RMSE), 20.45% (MAE) and 21.83% (MAPE) relative improvement in DidiSY dataset, 46.08% (RMSE), 38.44% (MAE) and 51.72% (MAPE) relative improvement in BikeNYC dataset, 56.48% (RMSE), 49.39% (MAE) and 48.51% (MAPE) relative improvement in TaxiBJ dataset. When comparing to the best state-of-the-art methods, our model still achieves 0.88% (RMSE) and 1.77% (MAE) relative improvement in DidiSY dataset, 2.95% (RMSE), 1.855% (MAE) and 5.80% (MAPE) relative improvement in BikeNYC dataset, 1.18% (MAE) and 2.13% (MAPE) relative improvement in TaxiBJ dataset.

4.3 Evaluation on multi-step prediction

Next, we evaluate the ability of our model in conducting multi-step prediction. We predict the passenger demand for the next three time steps and compare it to three methods: HA, ConvLSTM, and DCRNN. RMSE and MAE are used as comparison metrics. As can be observed from Figure 4, the prediction of HA remains the same for all time steps while the prediction of other methods deteriorates with time. Moreover, the performance of HA is the worst in all steps. In summary, our model performs the best for all steps and it deteriorates slower than the other two state-of-the-art methods.

5 CONCLUSIONS

In this paper, we studied the citywide multi-step passenger demand prediction problem. We proposed a new deep learning model based on the Graph Convolution Network and Long-Short Term Memory network under the encoder-decoder framework. Our proposed model can capture more accurate spatial-temporal correlations hidden in citywide historical passenger demand data, achieves multi-step prediction with less deterioration and is flexible to different city partitioning methods. Experimental results show that our model consistently outperforms all the baselines and discriminative state-of-the-art methods.

REFERENCES

[1] Thomas N Kipf and Max Welling. 2017. Semi-supervised Classification with Graph Convolutional Networks. In *ICLR'2017*.
[2] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion Convolutional Recurrent Neural Network: Data-driven traffic forecasting. In *ICLR*.
[3] Aditya Krishna Menon and Young Lee. 2017. Predicting short-term public transport demand via inhomogeneous Poisson processes. In *CIKM'2017*. ACM, 2207–2210.
[4] Shi Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NIPS'2015*. 802–810.
[5] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. In *AAAI'18*.
[6] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction.. In *AAAI'17*. 1655–1661.
[7] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. 2016. DNN-based prediction model for spatio-temporal data. In *SIGSPATIAL'2016*. ACM, 92.
[8] Kai Zhang, Zhiyong Feng, Shizhan Chen, Keman Huang, and Guiling Wang. 2016. A framework for passengers demand prediction and recommendation. In *2016 IEEE International Conference on Services Computing (SCC)*. IEEE, 340–347.