Origin-Destination Matrix Prediction via Graph Convolution: a New Perspective of Passenger Demand Modeling

Yuandong Wang
Beihang University
China
wangyd@act.buaa.edu.cn

Hongzhi Yin*†
The University of Queensland
Australia
h.yin1@uq.edu.au

Hongxu Chen
The University of Queensland
Australia
hongxu.chen@uq.edu.au

Tianyu Wo*
Beihang University
China
woty@act.buaa.edu.cn

Jie Xu
University of Leeds
United Kingdom
J.Xu@leeds.ac.uk

Kai Zheng
University of Electronic Science and Technology, China
zhengkai@uestc.edu.cn

ABSTRACT

Ride-hailing applications are becoming more and more popular for providing drivers and passengers with convenient ride services, especially in metropolises like Beijing or New York. To obtain the passengers' mobility patterns, the online platforms of ride services need to predict the number of passenger demands from one region to another in advance. We formulate this problem as an Origin-Destination Matrix Prediction (**ODMP**) problem. Though this problem is essential to large-scale providers of ride services for helping them make decisions and some providers have already put it forward in public[1], existing studies have not solved this problem well. One of the main reasons is that the ODMP problem is more challenging than the common demand prediction. Besides the number of demands in a region, it also requires the model to predict the destinations of them. In addition, data sparsity is a severe issue. To solve the problem effectively, we propose a unified model, Grid-Embedding based Multi-task Learning (**GEML**) which consists of two components focusing on spatial and temporal information respectively. The Grid-Embedding part is designed to model the spatial mobility patterns of passengers and neighboring relationships of different areas, the pre-weighted aggregator of which aims to sense the sparsity and range of data. The Multi-task Learning framework focuses on modeling temporal attributes and capturing several objectives of the ODMP problem. The evaluation of our model is conducted on real operational datasets from UCAR and Didi[2]. The experimental results demonstrate the superiority of our GEML against the state-of-the-art approaches.

[1]https://biendata.com/competition/UAI/
[2]They are ride-hailing services providers in China.

* Corresponding author.
† Contributing equally with the first author.

KDD '19, August 4–8, 2019, Anchorage, AK, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330877

CCS CONCEPTS

• **Information systems → Data mining**; *Enterprise applications*; *Spatial-temporal systems*; • **Computing methodologies → Neural networks**;

KEYWORDS

Demand Prediction; Graph Convolution; Multi-task Learning

ACM Reference Format:
Yuandong Wang, Hongzhi Yin*†, Hongxu Chen, Tianyu Wo*, Jie Xu, and Kai Zheng. 2019. Origin-Destination Matrix Prediction via Graph Convolution: a New Perspective of Passenger Demand Modeling. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330877

1 INTRODUCTION

Recently, ride-hailing applications are becoming prevalent choices of daily commuting, such as Didi, UCAR, and Uber, which aim to provide passengers with convenient ride services and improve the efficiency of public transportation. For instance, on Didi, millions of taxi-calling transactions are generated in Beijing per day [20]. To provide high-quality services and achieve company profits, ride-hailing platforms need to fully understand the passenger demands in real time. On one hand, to satisfy passenger demands, the platforms must possess the awareness of passengers' timely mobility patterns to pre-assign orders to service vehicles in advance. On the other hand, it is crucial to maximizing the profit by uncovering popular and high-profit routes from the historical passenger demands, thus avoiding empty drives (i.e., driving without passengers). Therefore, instead of merely forecasting the possible number of passenger demands within a region, it is rather important to gain knowledge of passenger demands in terms of the origin and destination of each trip. Because the demand quantity between two regions at different time slots not only carries the strength of passenger demands but also establishes a bootstrap to mine useful mobility patterns. Fortunately, with adequate modeling strategies and the availability of large-scale passenger transactions, passenger demands coupled with mobility patterns are becoming predictable.

In this paper, we investigate the modeling of passenger demand from a new perspective, which is defined as Origin-Destination

Matrix **P**rediction (**ODMP**). In summary, the origin-destination matrix carries two aspects of information: (1) combinations of different origins and destinations; and (2) the number of passenger demands for each origin-destination combination. The target of ODMP is to predict the number of ride-hailing orders from one geographical region to another in a given time slot.

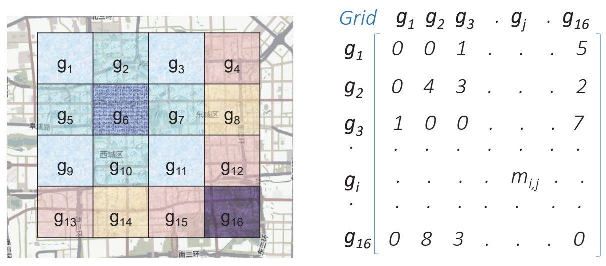
Unfortunately, in spite of existing studies on passenger demand modeling, the results achieved are hardly satisfactory. Recent studies [1, 17, 28, 29] attempt to predict the passenger demands by modeling the trajectory data acquired from GPS devices on taxies. However, the trajectory data cannot reveal passengers' exact mobilities as the onboard GPS only records the routes traveled by the vehicle, which contain many empty drives. To address this issue, passengers' mobility records (i.e. orders) from ride-hailing applications have been used in some recent works [20, 22, 24], which can reflect the real passenger demands and mobilities. Nevertheless, these methods only focus on the start points of orders and predict the quantity of orders arising in a region. In addition, these methods depend on auxiliary features extracted from multi-source data which are usually unavailable in most cases, thus limiting the generalizability of such models. Although prior works [12, 27] have considered the quantities of both in-flow and out-flow crowds in a certain area, they are incapable of matching the origins with exact destinations among the large volume of transactions.

Passengers' timely mobility pattern is a crucial factor for the effectiveness of demand modeling. For instance, in the morning rush hour, a large collection of crowds tend to travel from the residential areas to their workplaces. If we can discover such trends earlier and predict the passenger demands with origin-destination information, the ride-hailing platforms will be able to accurately suggest the possible dispatch location for each service vehicle in advance. However, it is quite challenging to model the passengers' mobility patterns and accurately predict the origin and destination at the same time due to the following reasons. First, it naturally requires to simultaneously consider (1) **the quantity of passenger demands** from a given region and (2) **the final destinations of these demands**. Most demand prediction problems are treated as simple regression tasks, i.e., approximating the quantity of ride services needed for a specific area in the recent future. However, this neither helps service providers understand their customers' desired destination, nor provides insights on popular routes. In real-life scenarios, it is crucial to not only estimate the number of passenger demands in a certain region but also uncover the corresponding destinations of these demands, thus offering more opportunities for the service providers to allocate their resources and improve their customer services for a specific area. The second challenge arises from **Spatial and Temporal feature fusion**. ODMP problems are associated with both spatial and temporal information, and the characteristics between different regions at different time slots vary significantly. It is hard for a model to model these two kinds of information in a unified way to effectively capture their dynamic correlation patterns. Last but not least, **Data Sparsity** is severe in ride-hailing records. The situation is quite common that there might be thousands of demands in particular urban areas, while some suburbs might only receive a few demands at the same time.

To this end, we propose a model called **G**rid-**E**mbedding based **M**ulti-task **L**earning (**GEML**) to directly model passengers' mobility records on graphs. Specifically, we represent passengers' origin-destination records associated with geographical regions in a graph where the nodes represent geographical regions (defined as Grids) and the links between nodes denote the existence of passenger demands, with the weights on links denoting the quantity of orders. Note that the grids are obtained by partitioning the whole map of an area (state, city, or town) at a different level of granularity, and the formal definition of a grid is provided in Section 2.2. With the defined grids, the OD matrix in a given time slot can be constructed. As shown in Figure 1, the area is partitioned into 16 grids, and the mobility records are summarized in the corresponding OD matrix.

GEML is inspired by recent advances in modeling and performing convolutions on graphs [10, 11]. The aggregation function defined in *Graph Convolutional Networks* (GCNs) can be regarded as a kind of message passing scheme [6, 9], which effectively captures the information flow into a node from its directly connected neighbors. Motivated by the message passing and neighborhood aggregation, the traffic flow from one grid to another can be modeled through mimicking the messages passed between the grids, and the embeddings of corresponding grids can be learned by aggregating the features of the connected grids. However, if we straightforwardly apply existing GCNs to the graph defined by the OD matrix, the learned embeddings for the grids with rare orders would tend to be unreliable and ineffective due to the data sparsity. Moreover, it would be infeasible to learn embeddings for isolated nodes (e.g., a newly built community) without any order history regardless of being as origins or destinations. To alleviate the sparsity issue of data, we propose to exploit the geographical correlation of grids based on the first law of geography [8] that everything is related to everything else but nearby things are more related than distant things. For example, the numbers of passenger demands in two geographically close grids tend to be proximate to each other. Specifically, we consider two kinds of neighbors in our grid embedding part and they are **Geographical Neighbors** and **Semantic Neighbors** based on whether two grids are geographically close or connected by passenger demands. The former one is used to measure the intrinsic closeness between one grid and its neighbors, while the semantic neighbors are used to model the semantic strength of the traffic flow between the origins and destinations in the grid network.

Based on the representation of each grid learned by grid embedding, we design a multi-task neural network for ODMP by incorporating the significant temporal information of passenger demands. Inspired by existing work that separately models the inbound and outbound traffic of a grid, we also conduct two subtasks which predict the numbers of specific incoming and outgoing demands in each grid at different time slots. The rationale for introducing these two subtasks is that we are able to capture more dynamic mobility patterns at each grid individually. With the supplement of two individual subtasks, the overall demand prediction task can capture stronger intrinsic temporal patterns because the overall demands within each grid have a substantially larger scale or granularity. For example, during the morning rush hours, the ride-hailing demands may have different destinations in terms of fine-granularity grids, resulting in the data sparsity issue, by that, we mean that the destinations of passenger demands may spread very widely, but



(a) The Grids (b) The OD Matrix

Figure 1: Example of Grids and OD Matrix

the aggregated incoming and outgoing demands for these grids are sufficient.

The main contributions of this work are summarized as follows:

- We propose a novel problem ODMP that predicts the passenger demands of given origins and destinations at a given time slot in a novel and unified way, which can significantly help ride-hailing platforms prepare cars and dispatch orders.
- We formulate the ODMP problem by dividing the interested area into grids on the map. Then we design the grid embedding network to perform embedding for each grid via graph convolutions among novelly defined grid neighborhoods (geographical and semantic neighbors), which models traffic transferring relationships among different grids by mimicking the message passing schema in GCNs.
- We design a Multi-task Learning network resorting to *Long Short-Term Memory Recurrent Networks* (LSTMs) for capturing temporal trends of passenger demands. Two subtasks predict individual incoming and outgoing demands in a grid, while the main task predicts the demands between each pair of grids.
- Extensive experiments on two real-world and large-scale ride-hailing datasets demonstrate the proposed GEML model outperforms baselines.

2 PRELIMINARIES

2.1 Definitions

We first introduce several fundamental concepts to formulate the ODMP problem. We assume the area of interest is partitioned into subareas as grids.

Definition 2.1. ***Grid.*** The entire spatial region of interest (such as a specific city) is divided into n non-overlapping grids, denoted by $G = \{g_1, g_2, ..., g_n\}$. Figure 1(a) depicts an example of Grids, where the region is partitioned into 16 grids. The range of every grid is defined by the maximum and minimum coordinates.

Note that we choose this method because it is general and handy. Some studies use the road network to divide the city area but road network data is not always completed or available in every city. Some studies just take the POIs as the origins and destinations, which would aggravate the sparsity of data. Because POIs are too microcosmic and enormous in scale, which makes them incapable of showing mobility patterns in a suitable granularity.

Definition 2.2. ***Time Slot.*** We evenly partition the time into a sequence of t slots, which are represented as $\{Slot_1, Slot_2, ..., Slot_t\}$. The interval between any two consecutive slots is constant.

Definition 2.3. ***OD Matrix.*** In each time slot, for any two grids $g_i, g_j \in G$, the total number of travel demands from g_i to g_j is denoted as $m_{i,j}$. Hence, we define the Origin-Destination (OD) Matrix as $\mathbf{M} \in \mathbb{N}^{G \times G}$ where each entry $m_{i,j} \in \mathbf{M}$ represents the number of passenger demands from grid g_i to grid g_j.

2.2 Origin-Destination Matrix Prediction

Problem 1. ***Origin-Destination Matrix Prediction.*** *For t time slots, given a sequence of t observed OD matrices $\{\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_t\}$ and a set of auxiliary features X_{aux} (optional based on availability), the Origin-Destination Matrix Prediction (**ODMP**) is a regression task to predict the OD Matrix $\mathbf{M}_{t+1}$ in the next time slot $Slot_{t+1}$.*

3 SOLUTION

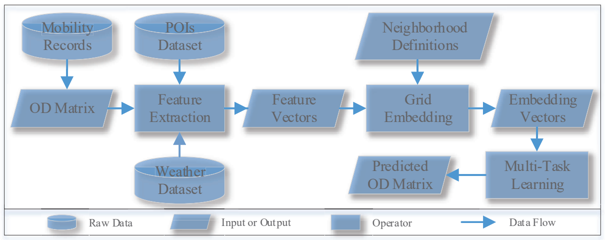


Figure 2: An overview of the GEML model.

In this section, we introduce the unified model Grid-Embedding based Multi-task Learning (**GEML**) which is a general solution to the ODMP problem. Figure 2 presents an overview of GEML. Firstly, based on Definition 4, the OD matrices can be extracted from the mobility records of ride service providers. External data sources will be used to produce auxiliary features if available. Then, the grid embedding part of GEML learns the embedding vector for each grid through the information aggregation of two types of its neighbors: geographical and semantic neighbors. Afterwards, the sequential vector representations of each grid will be fed into the multi-task neural network to learn a representation of a grid in the most recent time slot t_a. Finally, the vector representation of the grids will be utilized to generate the predicted OD matrix.

In our proposed GEML model, we capture both spatial and temporal patterns in a unified way. From the spatial perspective, we propose a neighbor-based grid embedding method to learn the vector representation of each grid by aggregating the information of its neighbors. From the temporal perspective, we design a multi-task learning framework to model the dynamic trends of passenger demands over time. In what follows, we will present the technical details in both grid embedding and multi-task learning.

3.1 Grid Embedding

Due to the limitation of GCNs on grids with low-scale demands, we propose two kinds of neighborhood functions under the context of ODMP for passenger demand modeling, namely geographical neighbors and semantic neighbors. They are utilized to measure the intrinsic closeness between one grid and its neighbors and capture the semantic strength of the traffic flow between the origin and destination in the grid network, respectively. Figure 3 depicts the aggregation of these two kinds of neighbors.

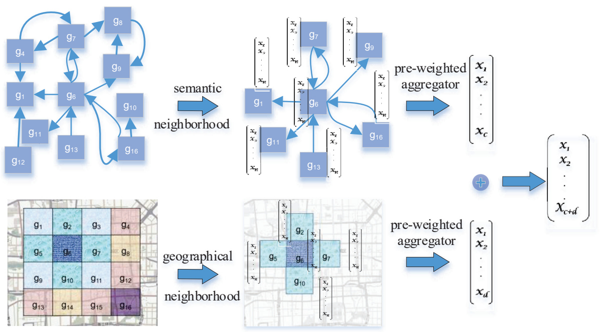


Figure 3: Illustration of grid embedding for both geographical and semantic neighborhoods.

3.1.1 ***Geographical Neighborhood.*** As Section 2.1 states, we divide the target area into non-overlapping grids. Geographical neighbors of grid g_i are those grids that are geographically adjacent with it. For a grid g_i, its geographical neighbor set is formulated as:

$$\Phi_i = \{g_j | dis(g_i, g_j) \leq L\}. \quad (1)$$

where $dis(g_i, g_j)$ denotes the spatial distance of these two grids' centers, the straight length from g_i's center to g_j's geographically. And L is a threshold of the distance which can determine the range of neighborhood. For instance, the distances between g_6 and g_1, g_2 in Figure 1(a) are $\sqrt{2}u$ and $1u$ respectively, where u denotes an arbitrary measurement unit. A grid's geographical neighbors are important for the modeling of itself. Intuitively, being the geographical neighbors of g_i, the grids in the set Φ_i are more likely to have close densities of passenger demands and share similar features with g_i. For example, if grid g_i and $g_j \in \Phi_i$ are adjacent and located in a sparsely populated suburb, then both of them are likely to have fewer demands.

3.1.2 ***Semantic Neighborhood.*** The major advantage of predicting passenger demand via OD matrix is that we can better discover the mobility relationships among different grids in a timely manner. If there is at least one demand between g_i and g_j (can be either direction), then they are the semantic neighbor of each other. For grid g_i, within an arbitrary time slot $t' = 1, 2, ..., t$, we can obtain a set of its semantic neighbors via:

$$\Omega^i_{t'} = \{g_j | m_{i,j} > 0 \parallel m_{j,i} > 0, m_{i,j} \in M_{t'}, m_{j,i} \in M_{t'}\}. \quad (2)$$

According to Eq.(2), it is worth mentioning that the numbers of different grids' semantic neighbors at different time slots are uncertain. The rationale of defining the semantic neighbors of each grid is that the characteristics of each grid are not only determined by the geographical neighbors surrounding it but also affected by its interaction patterns. Specifically, the interaction patterns of grid g_i in time slot t_k is reflected by other grids $g_j \in G$ that appears in the same trip with g_i (i.e., either from g_i to g_j or from g_j to g_i). Because the ODMP problem is time-sensitive, it is crucial to consider the semantic relationship between different grids at different time slots. For instance, large amounts of people tend to travel from remote areas to their workplaces in the city center during the morning rush hour. By introducing the concept of semantic neighbors, we can thoroughly take such timely patterns into account for the grid embedding learning.

3.1.3 ***Pre-Weighted Aggregator for Grid Embedding.*** In our model GEML, we infer the vector representation of each grid g_i in time slot t_k by aggregating the information of its geographical neighbors Φ_i and semantic neighbors $\Omega^i_{t'}$. Instead of training a distinct embedding vector for each grid, we train an aggregator function that learns to accumulate and select feature information from a grid's neighborhood. Before detailing our pre-weighted aggregator for grid embedding, we first briefly introduce the naive form of aggregator adopted by [10]:

$$\mathbf{v}_i = \sigma\Big(\mathbf{W} \cdot MEAN\big(\{\mathbf{v}'_i\} \cap \{\mathbf{v}'_j, g_j \in N_i\}\big)\Big), \quad (3)$$

where $\mathbf{v}_i$ is the embedding vector of grid g_i; $\mathbf{W}$ is the weight matrix to learn; σ is the nonlinear *Sigmoid* activation function; and $MEAN(\cdot)$ denotes the element-wise mean operator. The naive aggregation approach computes the grid feature $\mathbf{v}_i$ by taking the element-wise mean of the features $\mathbf{v}'_j$ of all its neighbors $g_j \in N_i$ and concatenating them to the previous feature $\mathbf{v}'_i$ of itself. However, despite some variants of the basic aggregator (e.g., pooling aggregator and LSTM aggregator [10]), existing aggregation methods in graph convolution lack sufficient ability to fully capture the relationship among different grids in the scenario of ODMP for demand modeling. The reason is that these aggregators are not able to differentiate the importance of each grid neighbor when fusing all their features. Intuitively, the closer the geographical distance between two grids is, the more similar attributes they will have. Also, in the semantic neighbor set, the popularity degree of a neighbor grid should pose influence on the aggregation process as it retains representative mobility patterns.

In light of this, we present a pre-weighted aggregator which can selectively lay more emphasis on important grid neighbors for grid embedding. For the geographical neighbors Φ_i of grid g_i, we leverage the distance between g_i and $g_j \in \Phi_i$, denoted by $dis(g_i, g_j)$ as a weighting factor for the aggregator. Accordingly, we formulate the pre-weighted aggregator for geographical neighbors as the following:

$$\mathbf{r}^i_{t'} = \sigma\Big(\mathbf{W}_g \cdot (\mathbf{f}^i_{t'} + \sum_{g_j \in \Phi_i} \frac{dis(g_i, g_j)}{\sum dis(g_i, g_j)} \mathbf{f}^j_{t'})\Big), \quad (4)$$

where $\mathbf{r}^i_{t'}$ denotes the geographical embedding vector of grid g_i at time t'; $\mathbf{W}_g$ is the trainable weight matrix; while $\mathbf{f}^i_{t'}$ and $\mathbf{f}^j_{t'}$ are respectively the features of g_i and $g_j \in \Phi_i$ before the geographical aggregation operation. Similarly, at each time t', we conduct the pre-weighted feature aggregation regarding the semantic neighbors $\Omega^i_{t'}$ of grid g_i:

$$\mathbf{s}^i_{t'} = \sigma\Big(\mathbf{W}_s \cdot (\mathbf{f}^i_{t'} + \sum_{g_j \in \Omega^i_{t'}} \frac{degree(g_j)}{\sum degree(g_j) + \epsilon} \mathbf{f}^j_{t'})\Big), \quad (5)$$

where the weighting factor $degree(g_j)$ is the degree of grid g_j (i.e., the number of demands starting from or ending at g_j during each time slot t'). In addition, ϵ is a very small value close to zero in case $degree(g_j) = 0$; $\mathbf{s}^i_{t'}$ denotes the semantic grid embedding of grid g_i at time t'; $\mathbf{W}_s$ is the weight matrix; while $\mathbf{f}^i_{t'}$ and $\mathbf{f}^j_{t'}$ are respectively the features of g_i and $g_j \in \Omega^i_{t'}$ before the semantic

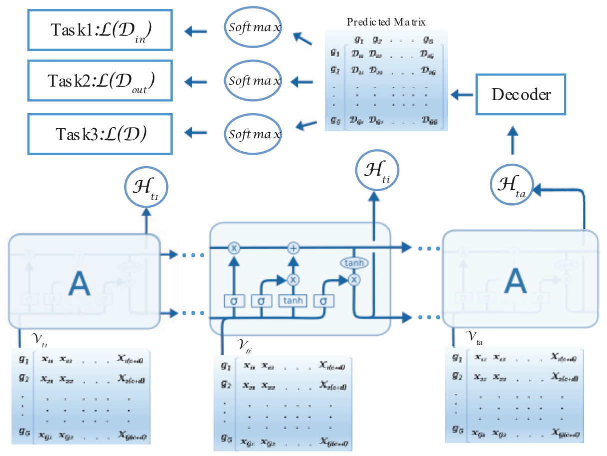


Figure 4: The architecture of the multi-task LSTM.

aggregation operation. It is worth noting that $\mathbf{r}^i_{t'}$ and $\mathbf{s}^i_{t'}$ are both updated for every successive time slot t' because the features for every grids and semantic neighbors for g_i are time-dependent.

Intuitively, the embedding vectors $\mathbf{r}^i_{t'}$ and $\mathbf{s}^i_{t'}$ carry the information of the geographical and semantic aspects respectively, which are learned in parallel via two distinct neighborhood contexts and feature spaces. In the last stage of grid embedding, the final representation $\mathbf{v}^i_{t'}$ of grid g_i at time t' is computed by combining the grid embedding vectors from both aspects:

$$\mathbf{v}^i_{t'} = [\mathbf{r}^i_{t'}, \mathbf{s}^i_{t'}], \tag{6}$$

where $[\cdot]$ represents the concatenation of two vectors.

3.2 Multi-Task Learning

With the final representation vectors, there will be a sequence of t embedding vectors for each grid g_i, i.e., $\{\mathbf{v}^i_1, \mathbf{v}^i_2, ..., \mathbf{v}^i_t\}$. In this section, we propose a multi-task learning scheme with periodic-skip LSTM for ODMP, whose architecture is shown in Figure 4. As a complementary to the main task, two subtasks enable our model to capture stronger intrinsic temporal patterns by enlarging the scale or granularity of overall demands within each grid.

3.2.1 *Periodic-Skip LSTM.*

Taking a time series $\{\mathbf{x}_t\}^T_{t=1}$ as input, *Recurrent Neural Network* (RNN) encodes $\{\mathbf{x}_t\}^T_{t=1}$ into hidden states $\{\mathbf{h}_t\}^T_{t=1}$ via $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$, where $f(\cdot)$ is a non-linear mapping function. To capture the long-range dependency [2, 16], we leverage the RNN with long short-term memory architecture (LSTM) via the following formulation [13]:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{7}$$

where $\sigma(\cdot)$ is the *Sigmoid* activation function; $\odot$ is the element-wise multiplication. $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$ and $\mathbf{c}$ are respectively the input gate, forget gate, output gate, and cell state vectors. When each of them is being updated, there are corresponding trainable weights $\mathbf{W}$ and

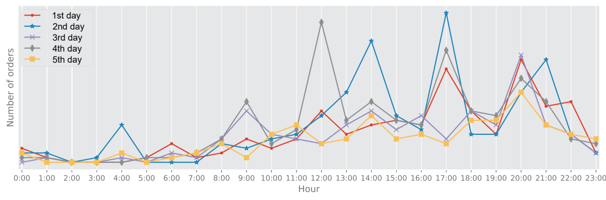


Figure 5: The number of passenger demands w.r.t. different hours and days.

the bias vectors $\mathbf{b}$. For notation convenience, we simplify the LSTM system in Eq.(7) as:

$$\mathbf{h}_t = LSTM(\mathbf{x}_t, \mathbf{h}_{t-1}). \tag{8}$$

Clearly, LSTMs can learn the representation of the current input based on its dependency on the hidden state learned from the previous time step, so $\mathbf{h}_t$ is actually more focused on inputs that are close to time t. However, this setting might not fit the scenario of ODMP-based demand modeling. To fully understand the dynamic patterns of passenger demands, we randomly sample 5 days' transactions from UCAR dataset (see Section 4) and plot the hourly passenger demands for each day in Figure 5. Apparently, at the same time of these days, the number of passenger demands share similar patterns. However, when predicting the passenger demands for the next hour, the sequence modeling scheme in LSTM will force the model to gather information from the previous consecutive hours. This may be less helpful for demand prediction as the irrelevant inputs incur much noise. In light of this, to better model the periodicity, we take the grid embedding sequence $\{\mathbf{v}^i_1, \mathbf{v}^i_2, ..., \mathbf{v}^i_t\}$ as inputs and further convert Eq.(8) into a periodic-skip LSTM which skips irrelevant sequential patterns:

$$\mathbf{h}^i_t = LSTM^{ps}(\mathbf{v}^i_t, \mathbf{h}^i_{t-p}) \tag{9}$$

where p is the number of hidden states skipped through. That is to say, the latent representation $\mathbf{h}^i_t$ of grid g_i is computed using the corresponding embedding $\mathbf{v}^i_t$ and the historical hidden states with a time step interval p.

3.2.2 *Main Task: Predicting the OD Matrix.*

By the periodic-skip LSTM framework, we can produce $\mathbf{h}^i_t$ which is the vector representation for grid g_i at time t. In order to obtain the value of each entry $m_{i,j}$ in the OD Matrix, we define a transition matrix $\mathbf{W}_m \in \mathbb{R}^{d\times d}$ to model the transition from grid g_i to g_j. By this mean, $\hat{m}_{i,j}$ (i.e., the predicted demand from g_i and g_j at $t+1$) can be computed as:

$$\hat{m}_{i,j} = (\mathbf{W}_m \mathbf{h}^i_t)^\top \mathbf{h}^j_t, \tag{10}$$

and we use mean squared error to compute the loss function for the main task:

$$\mathcal{L}_{ODMP} = \frac{1}{|\mathbf{M}_{t+1}| \times N} \sum_{n=1}^{N} ||\mathbf{M}_{t+1} - \hat{\mathbf{M}}_{t+1}||, \tag{11}$$

where $m_{i,j} \in \mathbf{M}_{t+1}$ is the real value in the OD matrix at time $t+1$, $n \le N$ is the index of the training sample.

3.2.3 *Two Subtasks: Predicting the In- and Out-Degrees.*

In parallel with the main task of predicting the overall OD Matrix described above, we also separately model the inbound and outbound

Table 1: A summary of datasets in use.

dataset	UCAR	Didi
time span	31 days	30 days
city	Beijing	Chengdu
total area	53×52 km^2	49×55 km^2
grid granularity	2.65×2.6km^2	3.27×3.67km^2
time slot granularity	1 hour	1 hour
auxiliary features	weather; POIs	-

traffic flow of a grid at a given time slot t. Eq.(12) and Eq.(13) work out the predicted number of orders going into (and out of) a grid.

$$\hat{p}_i = \mathbf{w}_{in}^{\top}\mathbf{h}_t^i, \tag{12}$$

$$\hat{q}_i = \mathbf{w}_{out}^{\top}\mathbf{h}_t^i, \tag{13}$$

where $\mathbf{w}_{in}$ and $\mathbf{w}_{out}$ are two projection weights that are used to project the grid embeddings to a scalar. Then, similar to the main task, the loss function for modeling the inbound and outbound orders can be formulated as:

$$\mathcal{L}_{IN} = \frac{1}{|G| \times N} \sum_{n=1}^{N} \sum_{g_i \in G} (p_{i,n} - \hat{p}_{i,n})^2. \tag{14}$$

$$\mathcal{L}_{OUT} = \frac{1}{|G| \times N} \sum_{n=1}^{N} \sum_{g_i \in G} (q_{i,n} - \hat{q}_{i,n})^2. \tag{15}$$

where **G** is the number of grids in the OD matrix, $n \leq N$ is the index of the training sample.

3.2.4 ***Loss Functions***. Corresponding to the three tasks defined above, we formulate the overall loss function by combining the losses for the main task as well as two subtasks:

$$\mathcal{L}_{GEML} = \eta\mathcal{L}_{ODMP} + \eta_{in}\mathcal{L}_{IN} + \eta_{out}\mathcal{L}_{OUT} \tag{16}$$

Because the importances of different loss functions may be different, η_{in}, η_{out}, η are added as the weights of them.

3.2.5 ***Optimization Strategy***. Above all, we optimize the parameters to minimize the final loss function defined in Equation (16). All parameters are updated by using the Stochastic Gradient Descent (SGD) method. Specifically, we use Adam [15], a variant of SGD to optimize the parameters in our model.

4 EVALUATION

This section evaluates GEML model through extensive experiments. Particularly, we aim to answer the following research questions:

RQ1 : How is the effectiveness of GEML on ODMP tasks?
RQ2 : How does each proposed component of the model contribute to the performance of GEML?
RQ3 : How does each major hyperparameter affect the prediction performance of GEML?
RQ4 : What are the actual mobility patterns learned by GEML?

4.1 Datasets

We conduct experiments on two real-world datasets generated by two ride-hailing applications, namely UCAR and Didi. The UCAR dataset is collected in Beijing urban area, from 1st to 31st of August in 2016. The Didi dataset ranges from 1st to 30th of November in 2016 and covers the urban area of Chengdu[3]. Table 1 summarizes the characteristics of two datasets. Note that both datasets have been desensitized. We divide Beijing and Chengdu into 400 and 225 grids respectively, based on grid granularities in Table 1, as it takes 5 minutes on average for a taxi driver to drive such distance, which is a reasonable waiting time for passengers as the statistics in [22]. The OD matrix sequences on both datasets are constructed with 1-hour granularity.

[3]https://outreach.didichuxing.com/research/opendata/

4.2 Baselines

To show the effectiveness of our GEML model on ODMP tasks, we compare GEML with the following state-of-the-art competitors.

- **HA:** We adopt History Average to predict each entry in the OD matrix with the mean of each grid's history demands.
- **LSTM:** LSTMs are designed to model long- and short-term dependencies, and are directly used in ODMP problems with different application scenarios. In this paper, we leverage LSTM with the settings in [14].
- **LSTNet:** LSTNet [16] is a state-of-the-art time series prediction model, which combines both LSTM and CNN for spatiotemporal feature modeling.
- **GCRN:** The recent proposed *Graph Convolutional Recurrent Network* (GCRN) [19] is built to model structured sequences. It combines CNNs on graphs with RNN to jointly identify spatial correlations and dynamic patterns.

To validate the performance gain from each component of our model, we implement five variants of GEML listed below:

- **GEML-S1:** We remove the aggregator for semantic neighbors and only keep the geographical aggregator.
- **GEML-S2:** We remove the subtasks of predicting out and in degrees of each grid and optimize the model via a single-task scheme which only predicts the OD matrix.
- **GEML-S3:** We replace the periodic-skip LSTM with a standard LSTM in Eq.(7).
- **GEML-S4:** We replace the pre-weighted aggregator with a naive mean aggregator in Eq.(3).
- **GEML-AF:** For UCAR dataset, we leverage the available auxiliary features of weather and POIs for GEML.

4.3 Experimental Settings

We evaluate the prediction accuracy with two widely-applied metrics, namely *Root Mean Square Error (RMSE)* and *Symmetric Mean Absolute Percent Error (SMAPE)*:

$$RMSE = \sqrt{\frac{1}{|\mathbf{M}_{t+1}| \times N} \sum_{n=1}^{N} ||\mathbf{M}_{t+1}^n - \hat{\mathbf{M}}_{t+1}^n||}, \tag{17}$$

$$SMAPE = \frac{2}{|\mathbf{M}_{t+1}| \times N} \sum_{n=1}^{N} \sum_{m \in \mathbf{M}_{t+1}^n} \frac{m - \hat{m}}{m + \hat{m} + 1}. \tag{18}$$

RMSE is scale-sensitive while SMAPE is not, so RMSE is suitable for the comparison within the same dataset, and SMAPE can be used to evaluate the same method across different datasets.

In order to check the effectiveness of our model on all the seven days of a week, we keep the last week of a month as the test set and the rest is training set, the last 10% of which is used for validation. To fairly compare model's capability, we train all models by optimizing

Table 2: Results on demand prediction of different methods.

Dataset	Metric	HA	LSTM	LSTNet	GCRN	**GEML**	GEML-S1	GEML-S2	GEML-S3	GEML-S4	GEML-AF
UCAR	RMSE	1.3515	1.3085	2.1559	0.3769	0.1605	0.2200	0.1791	2.5431	0.2281	0.2727
		-	+3.2%	-59.5%	+72.1%	+88.1%	+83.7%	+86.7%	-88.2%	+83.1%	79.8%
	SMAPE	0.7334	0.4141	1.1268	0.3083	0.0209	0.0403	0.0247	1.4310	0.0386	0.2165
		-	+43.5%	-53.6%	-53.6%	+97.2%	+94.5%	+96.6%	-95.1%	+94.7%	+70.5%
Didi	RMSE	2.8382	12.0027	19.5626	4.1512	1.6928	4.7939	2.5196	7.0960	2.0409	-
		0%	-322.9%	-589.3%	-46.3%	+40.3%	-68.9%	+11.2%	-150.0%	+28.1%	-
	SMAPE	0.6199	0.5121	1.3599	0.3842	0.2678	0.4960	0.4255	1.5883	0.3253	-
		-	+17.4%	-119.4%	+38.0%	+56.8%	+20.0%	+31.3%	-156.2%	+47.5%	-

the mean square error. The length of the skipped intervals p is set as 24 (i.e., one day). The weights of three sub-loss functions are set as $(0.25, 0.25, 0.5)$ while the number of the embedding dimension and hidden state are 128. All our models utilize the Mean Aggregator and include two layers. Because of the sparsity of the datasets, we use $L1$ regularization to enhance the weights' sparsity of our model. The learning rate is searched in [0.0001, 0.001, 0.01, 0.1], and batch size is 23. We realize our GEML model through Pytorch 0.4.1 on Python 2.7, which is based on the existing benchmark called GraphSAGE [10].

4.4 Performance Evaluation

4.4.1 ***The Effectiveness of GEML.*** Table 2 summarizes the results of all compared methods under RMSE and SMAPE. The percentage under the metric value represents the performance gain of each method with respect to HA. Note that all differences between our model and others are statistically significant ($p<0.01$).

From Table 2, we make the following observations:

- Unexpectedly, the results of LSTM and LSTNet on second dataset are obviously worse than HA. A possible reason for LSTM might be that LSTM is a totally time-series method which ignores the spatial attributes of the ODMP problem. While LSTNet utilizes CNN to model the spatial attributes. It is not suitable to OD Matrix because there is no meaningful spatial relationship between adjacent elements in a matrix.
- Methods tailored for temporal graph data (GCRN and GEML) achieve better overall performances. This might prove that it is more reasonable to model an OD Matrix as a graph when we solve the ODMP problem.
- The performance of HA is poor but stable on both datasets, which sometimes is even better than other baselines. This may enlighten us that if we can't catch the attributes of a problem, a simple model is a better choice.

As the auxiliary features of Didi dataset are not available, we just compare the rest four simple versions on this dataset. From the right columns of GEML in Table 2, it can be observed that:

- GEML-S3 has the worst performance on both datasets obviously, which testifies that our data owns strong periodicities among different days. The overall performance of GEML-S1 is the second worst one, and then GEML-S2, GEML-S4. Correspondingly, the periodic component plays the most important role to improve our model's performance.
- Out of our expectation, GEML-AF does not show any obvious advantage, and even worse than other simple versions. One possible reason might be that the two kinds of auxiliary features, POIs and weather do not belong to the same domain with the graph features. More precisely, our model is designed for the graph attributes, and these auxiliary features may become a disturbance to the model.

4.4.2 ***Analysis of Parameter Sensitivity.*** This section analyzes the parameter sensitivity of our model on the embedding dimension in Grid Embedding part and the weights of three loss functions in Multi-task Learning part.

Embedding Dimension. In Grid Embedding part a suitable setting of the embedding dimension is important for a sufficient representation. We conduct experiments on several alternatives of the embedding dimension, 32, 64, 128, 256 and 512. Figure 6 shows the results, from which, we can observe that:

- Didi's results fluctuate a lot with the increase of the embedding dimension while UCAR's behave more stably. This may be because of different sparsities and ranges of them. According to our statistics, Didi's dataset is denser, meanwhile, with a larger range than UCAR's.
- Both datasets obtain best results with the embedding dimension equaling to 128. Moreover, we can see that GEML obtains relatively fine results on UCAR when the embedding dimension is equal to 256. We suppose this is related to the feature dimension. UCAR's feature dimension is 802, almost twice of Didi's 452, so the embedding dimension 128 show a more obvious advantage on Didi while nearly even with the embedding dimension 256 on UCAR.

Weights of sub-loss functions. In Multi-task Learning part, we choose several sets of weights to conduct experiments. For example, the three values of each horizontal label e.g. $(0.1, 0.1, 0.8)$ in Figure 7 represent the weights of $\mathcal{L}_{IN}$, $\mathcal{L}_{OUT}$, and $\mathcal{L}_{ODMP}$ respectively. According to the results in Figure 7, we obtain the following observations:

- GEML still shows a more stable performance on UCAR's dataset rather than on Didi's under both metrics. It should be on account of the sparsity and scale of datasets.
- GEML performs best on Didi's dataset when the weights equal to $(0.25, 0.25, 0.5)$. This result seems reasonable because $\mathcal{L}_{IN}$ and $\mathcal{L}_{OUT}$ are designed to assist the prediction of OD Matrix, and the main target still should be minimizing $\mathcal{L}_{ODMP}$. The performance of GEML under different weights on UCAR is almost even. It is supposed to be related to the sparsity and range of the dataset, which makes this dataset not sensitive to the weights' variation.

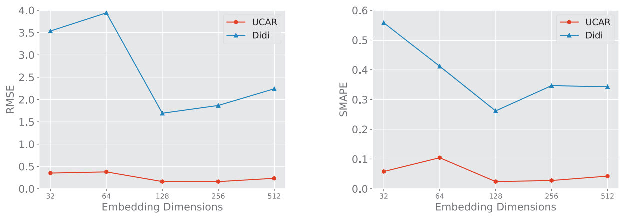


Figure 6: Results of different embedding dimensions

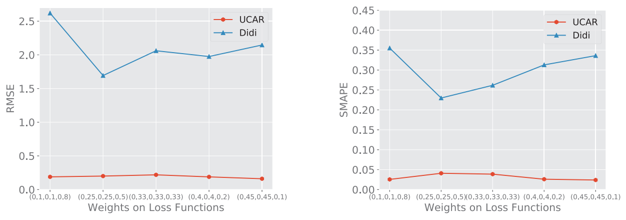


Figure 7: Results of different weights on loss functions

4.5 Visualization of Mobility Patterns

In order to give an intuitive presentation, Figure 8 shows a part of passengers' mobility patterns learned by GEML during different time slot on a work day in Beijing. We choose some transfers that contain more than 10 demands and get the integer of the predicted result by rounding it off. The red squares denote the *Grids* we have defined in Section 2 and the red arrows represent the passengers' transferring direction. The rectangles close to the arrows provide the number of transferred passengers between two grids. The first value in the rectangle is the predicted result of our model while the one in the bracket is its ground truth.

From Figure 8, we can draw the following observations:

- Figure 8(a) indicates that in the morning, people tend to leave home and march to different destinations, e.g. work or entertainment venues. Because many residential areas are located in g_{202}, g_{205}, g_{189} and g_{169}. Meanwhile, there is a software park located in the neighborhood of g_{267}, and g_{290} is the Olympic Forest Park.
- From Figure 8(b), we can see that after lunch, there are still many passengers leaving for work and entertainment venues. The famous Xidan Financial street just located in g_{168}, together with g_{167} and g_{189} which are residential and financial mixture areas. And near g_{253} and g_{233}, there is the largest art center in Beijing, well-known as the 798 Art District.
- According to Figure 8(c), there is no accordant direction for passengers' mobilities. For example, some people prefer to go home after an exhausting day from g_{129} to g_{272} which is almost a purely residential area. But even at 9:00 pm, there are still people to the 798 Art District. One possible reason may be that evening time is people's leisure time when people can choose any destinations as they like.

5 RELATED WORK

5.1 Demand and Mobility Prediction

In this subsection, we review mainstream schemes on the demand, mobility and other volume prediction problems.

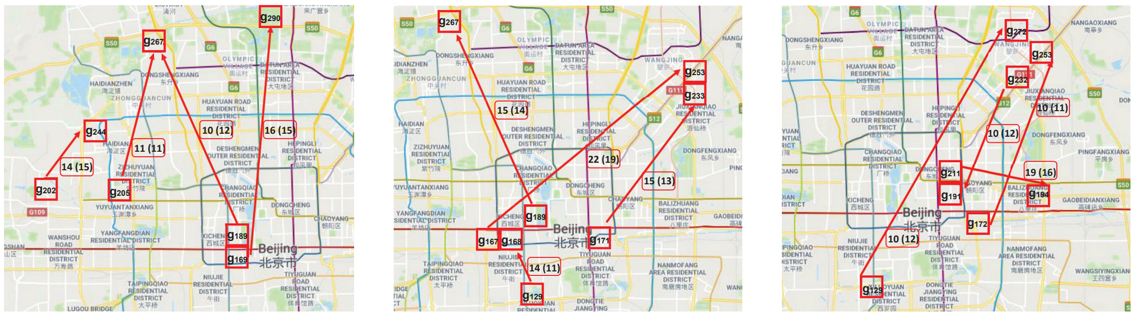
Studies of the common demand prediction only focus on origins of passenger demands. Some [28, 29] of them are based on trajectory data. Zhao et al. [29] employ a holistic approach to analyze both yellow cabs and Uber's trips in NYC and show high predictability of the taxi demand. Zhang et al. [28] propose a framework combining clustering and time-series forecasting based on historical taxi trajectories to predict taxi demands for hotspots in urban areas. The popularity of ride-hailing applications generates billions of passenger mobility records and some studies [20, 22, 24] conduct demand predictions based on this kind of data. Wei et al. [22] present a hybrid model to combine spatial features, the tendency of temporal fluctuation, and a classifier of zero-demand areas, which predicts integrated results for passenger demands of different areas. Tong et al. [20] put forward a unified linear regression model with more than 200 million dimensions of features extracted from multi-source data to predict the passenger demand for each POI. Yao et al. [24] propose a model consisting of temporal view, spatial view and semantic view, to predict the passenger demands within an area. However, all these methods ignore the destinations of passenger demands so that they cannot provide the global mobility patterns of passengers in a city.

There are some researches [7, 12, 14, 18, 27] studying passenger mobilities. Hoang et al. [12] integrate a seasonal model and a trend model, then a residual model to predict the numbers of in-flow and out-flow crowds of a region, while Zhang et al. [27] propose a deep-learning-based approach called ST-ResNet to solve the same task. But these two studies overlook the transferring relationships among areas. Ren and Xie [18] utilize the tensor decomposition to predict the OD trip matrix. Deng et al. [7] design a latent space model based on road networks to predict traffic matrix, which learns the attributes of vertices in latent spaces to capture both topological and temporal properties. LSTMs based methods are employed by [14]. Nevertheless, they either have not considered the problem from both spatial and temporal perspectives or fail to give an adequate and meaningful representation for each region. Chen et al. [5] propose a trend alignment with dual-attention multi-Task recurrent neural networks (TADA) to predict sales volume for supermarkets. Yin et al. [25] build a scalable probabilistic tensor factorization model (SPTF) for predicting heterogenous behavior data. But they are designed closely with the problem features, which is not suitable for our scenario.

5.2 Graph Representation Learning Based Methods.

Recently, some methods based on graph representation learning are proposed. Hamilton et al. [10] propose GraphSAGE, a general inductive framework that leverages node feature information to efficiently generate node embeddings for graph data. Unfortunately, they are just focused on the spatial perspective and cannot capture the temporal trend of the data. Seo et al. [19] build a model called GCRN to generalize the classical RNN to structured data by an arbitrary graph, which can be used to predict sequences of structured data. However, this method is incapable of modeling the transferring relationship between areas either because they have not taken the semantic neighbors into consideration.

There are some studies [3, 4, 21, 23, 26] inspiring us to model the passengers' mobilities into graph structures and consider different



(a) Mobility patterns at 7:00 o'clock (b) Mobility patterns at 14:00 o'clock (c) Mobility patterns at 21:00 o'clock

Figure 8: The visualization of mobility patterns on a workday in Beijing

perspectives of the problem. For example, Yin et al. [26] design a latent class statistical mixture model based on the observation that the behaviors of a user in social media systems, named temporal context-aware mixture model (TCAM), to account for the intentions and preferences behind user behaviors. Wang et al. [21] propose a streaming recommender model based on neural memory networks with external memories to capture and store both long-term stable and short-term dynamic interests in a unified way.

6 CONCLUSION

In this paper, we define ODMP problem of passenger demands, which can provide important references for drivers and ride service providers' decision-making. Comparing with the common demand prediction, it is more challenging because it requires to predict not only the number of demands in a area but also the destination of them. What's more, data sparsity is a severe issue in large-scale datasets. To address this problem, the first insight is to model the mobility patterns for areas through the informations of their neighbors, based on which we design the Grid-Embedding framework. And we add pre-weighted functions to its aggregating process so that it can sense the range and sparsity of the data. Then we utilize a multi-task LSTM with a periodic-skip component to model the temporal trend. The subtasks can assist the main task to capture stronger intrinsic temporal patterns by enlarging the scale or granularity of overall demands within each grid. The whole process is an end-to-end model called Grid-Embedding based Multi-task Learning (GEML). We evaluate our model on two real-world and large-scale ride-hailing datasets, which demonstrates the proposed GEML model outperforms baselines.

ACKNOWLEDGMENTS

The authors would like to thank Tong Chen for helping revise the paper over and over again. This work was supported by National Key R&D Program of China (2018YFB1004805), ARC Discovery Project (Grant No.DP190101985, DP170103954) and partially supported by NSFC (No.61836007, No.61832017, No.61532018).

REFERENCES

[1] Afian Anwar, Mikhail Volkov, and Daniela Rus. 2013. Changinow: A mobile application for efficient taxi allocation at airports. *ITSC* (2013).
[2] Yu-Wei Chao, Jimei Yang, Brian L Price, et al. 2017. Forecasting Human Dynamics from Static Images. *CVPR* (2017).
[3] Hongxu Chen, Hongzhi Yin, et al. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. *SIGKDD* (2018).
[4] Hongxu Chen, Hongzhi Yin, et al. 2019. Exploiting Centrality Information with Graph Convolutions for Network Representation Learning. *ICDE* (2019).
[5] Tong Chen, Hongzhi Yin, et al. 2018. TADA: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction. *ICDM* (2018).
[6] Hanjun Dai, Bo Dai, and Le Song. 2016. Discriminative embeddings of latent variable models for structured data. In *ICML*.
[7] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, et al. 2016. Latent space model for road networks to predict time-varying traffic. *SIGKDD* (2016).
[8] Stephen Gale and Gunnar Olsson. 2012. *Philosophy in geography*. Vol. 20. Springer Science & Business Media.
[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, et al. 2017. Neural message passing for quantum chemistry. *arXiv* (2017).
[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NIPS* (2017).
[11] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv* (2015).
[12] Minh X Hoang, Yu Zheng, and Ambuj K Singh. 2016. FCCF: forecasting citywide crowd flows based on big data. *SIGSPATIAL* (2016).
[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).
[14] Renhe Jiang, Xuan Song, Zipei Fan, et al. 2018. DeepUrbanMomentum: An Online Deep-Learning System for Short-Term Urban Mobility Prediction. *AAAI* (2018).
[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv* (2014).
[16] Guokun Lai, Wei-Cheng Chang, Yiming Yang, et al. 2018. Modeling long-and short-term temporal patterns with deep neural networks. *SIGIR* (2018).
[17] Luis Moreira-Matias, João Gama, Michel Ferreira, et al. 2012. A predictive model for the passenger demand on a taxi network. *ITSC* (2012).
[18] Jiangtao Ren and Qiwei Xie. 2017. Efficient OD Trip Matrix Prediction Based on Tensor Decomposition. *MDM* (2017).
[19] Youngjoo Seo, Michaël Defferrard, et al. 2018. Structured sequence modeling with graph convolutional recurrent networks. *ICONIP* (2018).
[20] Yongxin Tong, Yuqiang Chen, Zimu Zhou, et al. 2017. The simpler the better: a unified approach to predicting original taxi demands based on large-scale online platforms. *SIGKDD* (2017).
[21] Qinyong Wang, Hongzhi Yin, Zhiting Hu, et al. 2018. Neural Memory Streaming Recommender Networks with Adversarial Training. *KDD* (2018).
[22] Hua Wei, Yuandong Wang, Tianyu Wo, et al. 2016. Zest: a hybrid model on predicting passenger demand for chauffeured car service. *CIKM* (2016).
[23] Min Xie, Hongzhi Yin, Hao Wang, et al. 2016. Learning graph-based poi embedding for location-based recommendation. *CIKM* (2016).
[24] Huaxiu Yao, Fei Wu, Jintao Ke, et al. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. *arXiv* (2018).
[25] Hongzhi Yin, Hongxu Chen, et al. 2017. SPTF: a scalable probabilistic tensor factorization model for semantic-aware behavior prediction. *ICDM*.
[26] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. 2015. Dynamic user modeling in social media systems. *TOIS* (2015).
[27] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. *AAAI* (2017).
[28] Kai Zhang, Zhiyong Feng, Shizhan Chen, et al. 2016. A framework for passengers demand prediction and recommendation. *SCC* (2016).
[29] Kai Zhao, Denis Khryashchev, et al. 2016. Predicting taxi demand at high spatial resolution: approaching the limit of predictability. *Big Data* (2016).