Stochastic Origin-Destination Matrix Forecasting Using Dual-Stage Graph Convolutional, Recurrent Neural Networks

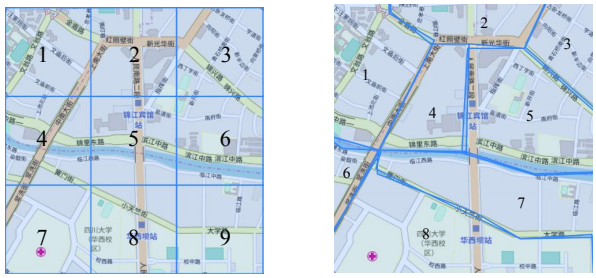
Jilin Hu[1], Bin Yang[1], Chenjuan Guo[1]✉, Christian S. Jensen[1], Hui Xiong[2]
[1]Department of Computer Science, Aalborg University, Denmark
[2]Management Science and Information Systems Department, Rutgers, the State University of New Jersey
{hujilin, byang, cguo, csj}@cs.aau.dk, hxiong@rutgers.edu

***Abstract*—Origin-destination (OD) matrices are used widely in transportation and logistics to record the travel cost (e.g., travel speed or greenhouse gas emission) between pairs of OD regions during different intervals within a day. We model a travel cost as a distribution because when traveling between a pair of OD regions, different vehicles may travel at different speeds even during the same interval, e.g., due to different driving styles or different waiting times at intersections. This yields *stochastic OD matrices*. We consider an increasingly pertinent setting where a set of vehicle trips is used for instantiating OD matrices. Since the trips may not cover all OD pairs for each interval, the resulting OD matrices are likely to be sparse. We then address the problem of forecasting *complete, near future* OD matrices from *sparse, historical* OD matrices. To solve this problem, we propose a generic learning framework that (i) employs matrix factorization and graph convolutional neural networks to contend with the data sparseness while capturing spatial correlations and that (ii) captures spatio-temporal dynamics via recurrent neural networks extended with graph convolutions. Empirical studies using two taxi trajectory data sets offer detailed insight into the properties of the framework and indicate that it is effective.**

I. Introduction

Origin-destination (OD) matrices are applied widely in location based services (LBSs) and online map services (e.g., transportation-as-a-service), where OD matrices are used for the scheduling of trips, for computing payments for completed trips, and for estimating arrival times [1], [2]. For example, Google Maps and ESRI ArcGIS Online offer OD matrix services to help developers build various LBSs. Further, increased urbanization contributes to making it increasingly relevant to capture and study city-wide traffic conditions. OD matrices may also be applied for this purpose.

To use OD-matrices, a city is partitioned into *regions*, and a day is partitioned into *intervals*. Each interval is assigned its own OD-matrix, and an element (i, j) in the matrix described the cost (e.g., travel speed, travel time, fuel consumption, or travel demand) of travel from region i to region j during the interval that the matrix represents. Different approaches can be applied to partition a road network, e.g., using a uniform grid or using major roads, as exemplified in Figure 1. In this paper, we focus on travel speed matrices. However, the proposed techniques can also be applied to other travel costs, such as travel time, fuel consumption, and travel demand.



(a) Grid-based Partition (b) Road-based Partition

Figure 1: Partition a City into Regions

As part of the increasing digitization of transportation, increasingly vast volumes of vehicle trip data are becoming available. We aim to exploit such data for composing OD matrices. Specifically, an element (i, j) of a speed matrix for a given time interval can be instantiated from the speeds observed in the trips that went from region i to region j during the relevant time interval.

We consider *stochastic OD matrices* where the elements represent uncertain costs by means of cost distributions [3] rather than deterministic, single-valued costs. The use of distributions models reality better and enables more reliable decision making [4], [5]. For example, element (i, j) may have a speed histogram $\{([10, 20), 0.5), ([20, 30), 0.3), ([30, 40), 0.2)\}$, meaning that the probability of traveling speed from region i to region j at 10-20 km/h is 0.5, at 20-30 km/h is 0.3, and at 30-40 km/h is 0.2. If passengers need to go from their home in region i to catch a flight in an airport in region j, and the shortest path from their home to the airport is 15 km, then we are able to derive a travel time (minutes) distribution: $\{(22.5, 30], 0.2), (30, 45], 0.3), (45, 90], 0.5)\}$. Therefore, the passengers need to reserve at least 90 minutes to avoid being late. If using only the average speed, thus to deriving an average travel time of 50 minutes, the passengers runs the risk of arriving too late.

We address the problem of *stochastic origin-destination matrix forecasting*—based on historical stochastic OD-matrices,

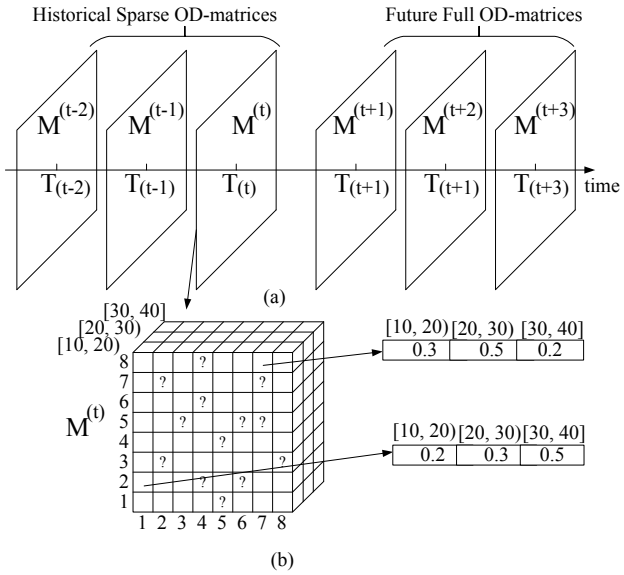


Figure 2: Stochastic Origin-Destination Matrix Forecasting

we predict future OD-matrices. Figure 2(a) shows an example: given stochastic OD-matrices for 3 historical intervals $T_{(t-2)}$, $T_{(t-1)}$, and $T_{(t)}$, we aim at predicting the stochastic OD-matrices for the 3 future intervals $T_{(t+1)}$, $T_{(t+2)}$, and $T_{(t+3)}$.

Here, a stochastic OD-matrix is represented as a 3-dimensional tensor $\mathbf{M}^{(t)}$, where the first dimension represents source regions, the second dimension represents destination regions, and the third dimension represents cost ranges. For example, Figure 2(b) shows the stochastic OD-matrix $\mathbf{M}^{(t)}$ for interval $T_{(t)}$, which is represented as an $\mathcal{R}^{8\times8\times3}$ tensor with 8 source regions, 8 destination regions, and 3 speed (km/h) ranges [10, 20), [20, 30), and [30, 40). Element (7, 8) in the OD-matrix is a vector (0.3, 0.5, 0.2), meaning that when traveling from region 7 to region 8, the travel speeds 10–20 km/h, 20–30 km/h, and 30–40 km/h have probabilities 0.3, 0.5, and 0.2, respectively.

Solving the stochastic OD-matrix forecasting problem is non-trivial as it is necessary to contend with two challenges. (1) **Data Sparseness.** To instantiate a stochastic OD-matrix for an interval using vehicle trips, we need to have sufficient trips for each region pair during the interval. However, even massive trip data sets are often spatially and temporally skewed [6], [7], making it almost impossible to cover all region pairs for all intervals. For example, the New York City taxi data set we use in experiments has 29+ million trips from November and December 2013. Yet, this massive trip set only covers 65% of all "taxizone" pairs in Manhattan, the most densely traversed region in New York City. If we further split the data set according to the temporal dimension, e.g., into 15-minutes intervals, the spareness problem becomes even more severe.

The data sparseness in turn yields sparse historical stochastic OD-matrices, where some elements are empty vectors with all zeros (e.g., the elements with "?" in Figure 2(b)). The challenge is how to use sparse historical OD-matrices to predict full future OD-matrices with no empty elements.

(2) **Spatio-temporal Correlations.** Traffic is often spatio-temporally correlated—if a region is congested during a time interval, its neighboring regions are also likely to be congested in subsequent intervals. Thus, to predict accurate OD matrices, we need to account for such spatio-temporal correlations. However, the OD matrices themselves do not necessarily capture spatial proximity. No matter which partition method is used, we cannot always guarantee that two geographically adjacent regions are represented by adjacent rows and columns in a matrix. For example, in Figure 1(a), regions 1 and 4 are geographically adjacent, but they are not adjacent in the OD matrices. In Figure 1(b), a similar case considers regions 4 and 7. This calls for a separate mechanism that is able to take into account the geographical proximity of regions that is well represented well in OD matrices. In addition, temporal dependency should be accounted for.

We propose a data-driven, end-to-end learning framework to forecast stochastic OD matrices while effectively addressing the challenges caused by data sparseness and spatio-temporal correlations. First, to address the *data sparseness* challenge, we factorize a sparse OD matrix into two small dense matrices with latent features of the source regions and the destination regions, respectively. To account for the *spatial proximity* of regions when factorizing the sparse OD matrix, we introduce two proximity matrices that model the spatial relationships among source regions and among destination regions, respectively. Then, we employ graph convolution to take into account the spatial relationships modeled by these two proximity matrices to enable the factorization. Second, to account for the spatio-temporal correlations, we integrate graph convolution that takes into account spatial relationships with recurrent neural networks that considers temporal dynamics on the two sequences of the small dense matrices. This enables accurate prediction that captures *spatio-temporal correlations*. Finally, we apply a multiplication to the two predicted, small dense matrices to obtain predicted full OD-matrices.

To the best of our knowledge, this is the first study of stochastic OD matrix forecasting that contends with data sparseness and spatio-temporal correlations. The study makes four contributions. First, it formalizes the stochastic OD matrix forecasting problem. Second, it proposes a generic framework to solve the problem based on matrix factorization and recurrent neural networks. Third, it extends the framework by embedding spatial correlations using graph convolutions into the factorization stage and the RNN stage. Fourth, it encompasses extensive experiments using two real-world taxi datasets that offers insight into the effectiveness of the framework.

II. Related Work

Travel Cost Forecasting: We consider three types of travel cost forecasting methods in turn: segment-based methods [8], [9], [10], [11], [12], [13], [14], [15], path-based methods [16], [17], [18], [19], [20], [21], [22], [23], and OD-based methods [6], [2], [24], [25].

Segment-based methods focus on predicting the travel costs of individual road segments. By modeling the travel costs of a road segment as a time series, techniques such as time-varying linear regression [8], Markov models [9], [10], neural networks [26], [27], and support vector regression [11] can be applied to predict future travel costs. Most such models consider time series from different edges independently, while the spatio-temporal Hidden Markov model [10] takes into account the correlations among the costs of different edges. Some other studies focus on estimating high-resolution travel costs, such as uncertain costs [12] and personalized costs [13].

Path-based methods focus on predicting the travel costs of paths. A naive approach is to predict the costs of the edges in a path and then aggregate the costs. However, this approach is inaccurate since it ignores the dependencies among the costs of different edges in paths [17] and the spatial context on road networks [28], [29]. Other methods [16], [17], [18] use sub-trajectories to capture such dependencies and thus to provide more accurate travel costs for paths. A few studies propose variations of deep neural networks [19], [20] to enable accurate path travel-time prediction.

Finally, OD-based methods aim at predicting the travel cost for given OD pairs. Our proposal falls into this category. In this setting, since only OD pairs are given, but not specific segments and paths, neither segment-based nor path-based methods can be applied directly. A simple method based on weighted average [6] and a more advanced method based on deep learning [30] are proposed. However, they do not address data sparseness, which means that if no data is available for a given OD pair, they cannot provide a prediction. In contrast, our proposal is able to predict full OD-matrices without empty elements based on historical, sparse OD-matrices. A recent study [2] utilizes deep learning with multi-task learning to estimate OD travel time while addressing data sparseness. However, that study only considers the daily/weekly temporal patterns, whole not the temporal feature in the near history. Thus, it cannot provide in time predictions, e.g., traffic accidents. Further, existing proposals support only deterministic costs, while our proposal also supports stochastic costs.

Graph Convolutional Neural Network: Convolutional Neural Networks (CNNs) have been used widely in many areas, including trajectory data [31], where the underlying data is represented as a matrix [32], [33]. For example, when representing an image as a matrix, nearby elements, e.g., pixels, share local features, e.g., represent parts of the same object. In contrast, in our setting, an OD-matrix may not satisfy the assumption that makes CNNs work—two adjacent rows in an OD matrix may represent two geographically distant regions and may not share any features; and two separated rows in an OD matrix may represent geographically close regions that share many features.

Graph convolutional neural networks (GCNNs) [32], [33], [34] aim to address this challenge. The geographical relationships among regions can be modeled as a graph, and GCNNs then take into account the graph while learning. However, most studies consider a setting where only one dimension needs to be modeled as a graph. In contrast, in our study, both dimensions, i.e., the source and destination region dimensions, need to be modeled as two graphs. A recent study focuses on so-called geomatrix completion which considers a similar setting where two dimensions need to be modeled as two graphs. It uses multi-graph neural networks [35] with RNNs. However, the RNNs in this study are utilized to perform iterations to approximate the geomatrix completion, not to capture temporal dynamics as in our study. To the best of our knowledge, our study is the first that constructs a learning framework involving dual stage graph convolution and employing RNNs to forecast the future.

III. Preliminaries

For a **trip** $\boldsymbol{p} = (o, d, t, l, \tau)$, o, d denote the origin and destination, t is the departure time, l represents the trip distance, and τ is the travel time of the trip. Given $\boldsymbol{p}.l$ and $\boldsymbol{p}.\tau$, we derive the average travel speed v of $\boldsymbol{p}$. We use $\mathbb{P}$ to denote a set of historical trips.

To capture time-dependent traffic, we partition the time domain into a set of intervals (TI). For example, one day can be partitioned into 96 15-minutes intervals. For each time interval $T_i \in TI$, we obtain the set of historical trips $\mathbb{P}_{T_i}$ from $\mathbb{P}$ whose departure times belong to time interval T_i, i.e., $\mathbb{P}_{T_i} = \{\boldsymbol{p_i} | \boldsymbol{p_i}.t \in T_i \wedge \boldsymbol{p_i} \in \mathbb{P}\}$.

We further partition a city into M regions $\mathbb{V} = \{\mathbb{V}_1, \cdots, \mathbb{V}_M\}$. An **Origin-Destination (OD) pair** is defined as a pair of regions $(\mathbb{V}_o, \mathbb{V}_d)$ where $1 \leq o, d, \leq M$.

Given a time interval T_i, two regions $\mathbb{V}_o$ and $\mathbb{V}_d$, we obtain a trip set $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d} = \{\boldsymbol{p}_i | \boldsymbol{p}_i.o \in \mathbb{V}_o \wedge \boldsymbol{p}_i.d \in \mathbb{V}_d \wedge \boldsymbol{p}_i.t \in T_i\}$, meaning that each trip in $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ starts from region $\mathbb{V}_o$, at a time in interval T_i, and ends at region $\mathbb{V}_d$.

Next, we construct an equi-width histogram $\mathbb{H}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ to record the stochastic speed of trips in $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$. In particular, an equi-width histogram is a set of K bucket-probability pairs, i.e., $\mathbb{H}_{T_i,\mathbb{V}_o,\mathbb{V}_d} = \{(b_j, pr_j)\}$. A bucket $b_j = [v_s, v_e)$ represents the speed range from v_s to v_e, and all buckets have the same range size. Probability pr_j is the probability that the average speed of a trip falls into the range b_j. For example, the speed histogram $\{([0, 20), 0.5), ([20, 40), 0.3), ([40, 60), 0.2)\}$ for $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ means that the probabilities that the average speed (km/h) of a trip in $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ falls into $[0, 20)$, $[20, 40)$, and $[40, 60)$ are 0.5, 0.3, and 0.2, respectively.

Given a time interval T_i, an **OD stochastic speed tensor** is defined as a matrix $\mathbf{M}^{(i)} \in \mathbb{R}^{N \times N' \times K}$, where the first and second dimensions range over the origin and destination regions, respectively, and the third dimension ranges over the stochastic speeds. For generality, the origin and destination regions can be the same or can be different; thus the first and second dimensions have N and N' instances, respectively. The third dimension defines K speed buckets. $M_{o,d,k}$ represents the cell (o, d, k) of tensor $\mathbf{M}^{(i)} \in \mathbb{R}^{N \times N' \times K}$ and represents the probability of trips in $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ traveling at an average speed that falls into the k-th bucket.

Following the example in Figure 2(b), given time interval T_i, for origin region 7 and destination region 8, we obtain

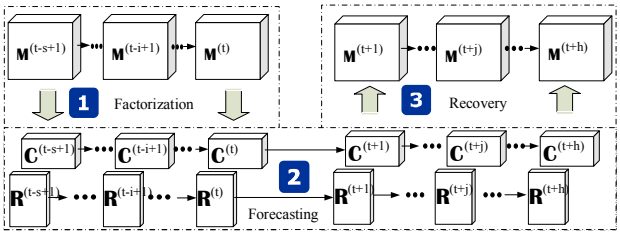


Figure 3: Framework Overview

a stochastic speed of trips as a histogram, in which the first bucket records that the probability of trips, starting at region 7 during time interval T_i and ending at region 8, traveling at speed $[10, 20)$ is 0.3.

As shown in Figure 2(b), not all cells have a histogram to capture the stochastic speed. Specifically, the cells with question marks have no histograms because no trip records are available for those cells, i.e., $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d} = \emptyset$, so that $\mathbb{H}_{T_i,\mathbb{V}_o,\mathbb{V}_d} = \emptyset$. We refer to such tensor as *sparse OD stochastic speed tensor*. Given a time interval T_i, we refer to a tensor where each cell has a stochastic speed $\mathbb{H}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ as a *full OD stochastic speed tensor*.

Problem Statement: Given s sparse OD stochastic speed tensors $\mathbf{M}^{(t-s+1)}$, ..., $\mathbf{M}^{(t)}$ in the past s historical time intervals $T^{(t-s+1)}$, ..., $T^{(t)}$, we aim to predict the stochastic speeds for the next h time intervals $T^{(t+1)}$, ..., $T^{(t+h)}$ in the form of h full OD stochastic speed tensors $\mathbf{M}^{(t+1)}$, ..., $\mathbf{M}^{(t+h)}$.

$$[\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}] \rightarrow [\mathbf{M}^{(t+1)}, \ldots, \mathbf{M}^{(t+h)}]$$

IV. Basic Forecasting Framework

A. Framework Overview and Intuition

Figure 3 shows the basic forecasting framework, which consists of three steps: *Factorization*, *Forecasting*, and *Recovering*. The basic framework contends with data sparseness in the factorization and recovering steps while capturing temporal dependency in the forecasting step. The pseudo code of the basic framework is shown in Algorithm 1. Spatial dependency is captured only in the advanced framework covered in Section V, not in the basic framework.

For the historical time intervals $T^{(t-s+1)}$, ..., $T^{(t)}$, we have sparse OD stochastic speed tensors $\mathbf{M}^{(t-s+1)}$, ..., $\mathbf{M}^{(t)}$. To contend with data sparseness, we factorize each tensor $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N\times N'\times K}$, $i \in [1, s]$, into two smaller, but dense tensors $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N\times\beta\times K}$ and $\mathbf{C}^{(t-i+1)} \in \mathbb{R}^{\beta\times N'\times K}$, where $\beta \ll N, N'$. We use two dense tensors for the factorization to support the case where the latent features of the origin and destination regions are different. Thus, $\mathbf{R}^{(t-i+1)}$ and $\mathbf{C}^{(t-i+1)}$ capture the hidden features of stochastic speeds among origin regions and among destination regions, respectively.

The factorization is supported by low-rank matrix approximation [35]. The intuition behind the factorization on origin and destination regions is two-fold. First, stochastic speeds among origin regions and among destination regions share hidden features, as traffic in a region affects the traffic in its nearby regions. Second, although the traffic between an OD region pair also affects travel speeds, we may not be able to take into account such information since the trip data, such as the New York taxi data, does not provide any information in-between the OD region pair.

Next, we consider $\mathbf{R}^{(t-s+1)}, \ldots, \mathbf{R}^{(t)}$ as an input sequence, from which we capture the temporal dependencies among the origin regions of $\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}$. We feed this input sequence into a sequence-to-sequence RNN model [36] to *forecast* an output sequence that represents the hidden features among the origin regions in the future. We apply a similar procedure to $\mathbf{C}^{(t-s+1)}, \ldots, \mathbf{C}^{(t)}$ to *forecast* an output sequence that represents the hidden features among destination regions in the future.

Finally, we *recover* $\mathbf{M}^{(t+j)}$ as a predicted, full OD stochastic speed tensor from $\mathbf{R}^{(t+j)}$ and $\mathbf{C}^{(t+j)}$, $j \in [1, h]$.

Algorithm 1 Basic Forecasting Framework

Input:
Sparse OD stochastic speed tensors: $\mathbf{M}^{(t-s+1)}$, ..., $\mathbf{M}^{(t)}$;
Output:
Full OD stochastic forecasts: $\widehat{\mathbf{M}}^{(t+1)}, \ldots, \widehat{\mathbf{M}}^{(t+h)}$;

Initialize RNN states in encoder: $\boldsymbol{H}_C^{(1)}, \cdots, \boldsymbol{H}_C^{(L)}$, and $\boldsymbol{H}_R^{(1)}, \cdots, \boldsymbol{H}_R^{(L)}$;
Initialize RNN states in decoder: $\hat{\boldsymbol{H}}_C^{(1)}, \cdots, \hat{\boldsymbol{H}}_C^{(L)}$, and $\hat{\boldsymbol{H}}_R^{(1)}, \cdots, \hat{\boldsymbol{H}}_R^{(L)}$;
for $i = t - s + 1, \cdots, t$ **do**
  $\mathbf{R}^{(0)} \leftarrow \mathrm{FC}_R(\mathbf{M}^{(i)})$, $\mathbf{C}^{(0)} \leftarrow \mathrm{FC}_C(\mathbf{M}^{(i)})$; (Section IV-B)
  for $l = 1, \cdots, L$ **do**
    $\mathbf{R}^{(l)} = \boldsymbol{H}_R^{(l)} \leftarrow \mathrm{GRU}_{enc}^R(\mathbf{R}^{(l-1)}, \boldsymbol{H}_R^{(l)})$;
    $\mathbf{C}^{(l)} = \boldsymbol{H}_C^{(l)} \leftarrow \mathrm{GRU}_{enc}^C(\mathbf{C}^{(l-1)}, \boldsymbol{H}_C^{(l)})$; (Section IV-C)
  end for
end for
for $i = t + 1, \cdots, t + h$ **do**
  $\widehat{\mathbf{R}}^{(0)} \leftarrow \boldsymbol{H}_R^{(L)}$, $\widehat{\mathbf{C}}^{(0)} \leftarrow \boldsymbol{H}_C^{(L)}$;
  for $l = 1, \cdots, L$ **do**
    $\widehat{\mathbf{R}}^{(l)} = \hat{\boldsymbol{H}}_R^{(l)} \leftarrow \mathrm{GRU}_{dec}^R(\widehat{\mathbf{R}}^{(l-1)}, \hat{\boldsymbol{H}}_R^{(l)})$;
    $\widehat{\mathbf{C}}^{(l)} = \hat{\boldsymbol{H}}_C^{(l)} \leftarrow \mathrm{GRU}_{dec}^C(\widehat{\mathbf{C}}^{(l-1)}, \hat{\boldsymbol{H}}_C^{(l)})$;
  end for
  $\widetilde{\mathbf{M}}^{(i)} \leftarrow \widehat{\mathbf{R}}^{(L)} \times (\widehat{\mathbf{C}}^{(L)})$ (See Eq. 2);
  $\widehat{\mathbf{M}}^{(i)} \leftarrow \mathrm{softmax}(\widetilde{\mathbf{M}}^{(i)})$ (See Eq. 3);
end for
if Is training. **then**
  Compute loss $L(\mathbf{M}, \widehat{\mathbf{M}})$ (See Eq. 4);
  Minimize the loss through back-propagation;
end if

B. Factorization

Given an input sparse OD stochastic tensors $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N\times N'\times K}$, we proceed to describe the method for factorizing

$\mathbf{M}^{(t-i+1)}$ into $\mathbf{R}^{(t-i+1)}$ and $\mathbf{C}^{(t-i+1)}$, which are able to capture the hidden features of stochastic speed among origin and destination regions, respectively. We first flatten $\mathbf{M}^{(t-i+1)}$ into a vector $\boldsymbol{f}^{(t-i+1)} \in \mathbb{R}^l$, where $l = N \cdot N' \cdot K$, from which we generate two small factorization vectors, $\boldsymbol{c}^{(t-i+1)} \in \mathbb{R}^{N' \cdot K \cdot \beta}$ and $\boldsymbol{r}^{(t-i+1)} \in \mathbb{R}^{N \cdot K \cdot \beta}$ using two fully-connected layers (FC). Here, β represents the dimension of the hidden features, which is a hyper-parameter to be set when learning.

Next, we reorganize the factorization vectors $\boldsymbol{r}^{(t-i+1)}$ and $\boldsymbol{c}^{(t-i+1)}$ into factorization tensors $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N \times \beta \times K}$ and $\mathbf{C}^{(t-i+1)} \in \mathbb{R}^{\beta \times N' \times K}$, respectively.

C. Forecasting

Given historical time intervals $T^{(t-s+1)}, \ldots, T^{(t)}$, we learn the temporal correlations of $\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}$ from the temporal correlations among origin regions $\mathbf{R}^{(t-s+1)}, \ldots, \mathbf{R}^{(t)}$ and the temporal correlation among destination regions $\mathbf{C}^{(t-s+1)}, \ldots, \mathbf{C}^{(t)}$.

Based on $\mathbf{R}^{(t-s+1)}, \ldots, \mathbf{R}^{(t)}$, we use a sequence-to-sequence RNN model [36] to forecast $\widehat{\mathbf{R}}^{(t+1)}, \ldots, \widehat{\mathbf{R}}^{(t+h)}$ for the future time intervals $T_{t+1}, \ldots, T_{t+h}$. In particular, we apply Gated Recurrent Units (GRUs) in the RNN architecture, since these can capture temporal correlations well by using gate units well and also offer high efficiency [37]. The process is presented as follows.

$$\widehat{\mathbf{R}}^{(t+1)}, \ldots, \widehat{\mathbf{R}}^{(t+h)} = seq2seqGRU(\mathbf{R}^{(t-s+1)}, \ldots, \mathbf{R}^{(t)}). \tag{1}$$

A similar procedure is applied to obtain $\widehat{\mathbf{C}}^{(t+1)}, \ldots, \widehat{\mathbf{C}}^{(t+h)}$ from $\mathbf{C}^{(t-s+1)}, \ldots, \mathbf{C}^{(t)}$.

D. Recovery

Given predicted tensors $\widehat{\mathbf{R}}^{(t+j)} \in \mathbb{R}^{N \times \beta \times K}$ and $\widehat{\mathbf{C}}^{(t+j)} \in \mathbb{R}^{\beta \times N' \times K}$ for a future time interval $T^{(t+j)}$, with $j \in [1, h]$, we proceed to describe how to transform $\widehat{\mathbf{R}}^{(t+j)}$ and $\widehat{\mathbf{C}}^{(t+j)}$ into a full OD stochastic speed tensor $\mathbf{M}^{(t+j)} \in \mathbb{R}^{N \times N' \times K}$.

First, we slice each of $\widehat{\mathbf{R}}^{(t+j)}$ and $\widehat{\mathbf{C}}^{(t+j)}$ by the speed bucket dimension into K matrices. Specifically, we have $slice(\widehat{\mathbf{R}}^{(t+j)}) = \{\widehat{\mathbf{R}}^{(t+j)}_{:,:,1} \cdots, \widehat{\mathbf{R}}^{(t+j)}_{:,:,K}\}$ and $slice(\widehat{\mathbf{C}}^{(t+j)}) = \{\widehat{\mathbf{C}}^{(t+j)}_{:,:,1} \cdots, \mathbf{C}^{(t+j)}_{:,:,K}\}$, where $\widehat{\mathbf{R}}^{(t+j)}_{:,:,k} \in \mathbb{R}^{N \times \beta}$ and $\widehat{\mathbf{C}}^{(t+j)}_{:,:,k} \in \mathbb{R}^{\beta \times N'}$, $k \in [1, K]$.

Next, we conduct a matrix multiplication as follows.

$$\widetilde{M}^{(t+j)}_k = \widehat{\mathbf{R}}^{(t+j)}_{:,:,k} \times (\widehat{\mathbf{C}}^{(t+j)}_{:,:,k}), \tag{2}$$

where $\widetilde{M}^{(t+j)}_k \in \mathbb{R}^{N \times N'}$, $k \in [1, K]$.

Finally, we are able to construct a tenor $\widetilde{\mathbf{M}}^{(t+j)} \in \mathbb{R}^{N \times N' \times K}$ by combining a total of K matrices, i.e., $\widetilde{\mathbf{M}}^{(t+j)}_{:,:,k} = \widetilde{M}^{(t+j)}_k$, $k \in [1, K]$. Now, $\widetilde{\mathbf{M}}^{(t+j)}$ is a full tensor where each element has a value.

A histogram $\widehat{\mathbf{M}}^{(t+j)}_{o,d,:} \in \mathbb{R}^{1 \times K}$ must meet two requirements to be a meaningful histogram: (1) $\widehat{\mathbf{M}}^{(t+j)}_{o,d,k} \in [0, 1]$, $k \in [1, K]$, meaning that the probability of a speed falling into the k-th bucket for each OD pair (o, d) must between 0 and 1; and (2) $\sum_{k=1}^{K} \widehat{\mathbf{M}}^{(t+j)}_{o,d,k} = 1$, meaning that the probability of a speed falling into all K buckets for each (o, d) must equal 1.

To achieve this, we apply a softmax function to normalize values in $\widetilde{\mathbf{M}}^{(t+j)}$ into $\widehat{\mathbf{M}}^{(t+j)}_{o,d,:}$ that satisfies the histogram requirements.

$$\widehat{\mathbf{M}}^{(t+j)}_{o,d,:} = softmax(\widetilde{\mathbf{M}}^{(t+j)}_{o,d,:}), \forall o \in [1, N], \forall d \in [1, N']. \tag{3}$$

Thus, we obtain h meaningful full OD stochastic speed tensors for the future time intervals $T_{t+1}, \ldots, T_{t+h}$ as the output of the recovery process: $\widehat{\mathbf{M}}^{(t+1)}, \ldots, \widehat{\mathbf{M}}^{(t+h)}$.

E. Loss Function

The loss function is defined as the error between the recovered future tensor and the ground-truth future tensor.

$$\ell = \sum_{j=1}^{h} [\lambda_{\mathbf{R}} ||\widehat{\mathbf{R}}^{(t+j)}||_F^2 + \lambda_{\mathbf{C}} ||\widehat{\mathbf{C}}^{(t+j)}||_F^2 + ||\Omega^{(t+j)} \circ (\mathbf{M}^{(t+j)} - \widehat{\mathbf{M}}^{(t+j)})||_F^2], \tag{4}$$

where $\lambda_{\widehat{\mathbf{R}}}$ and $\lambda_{\widehat{\mathbf{C}}}$ are the regularization parameters for $\widehat{\mathbf{R}}$ and $\widehat{\mathbf{C}}$, respectively. Further, $\Omega^{(t+j)} \in \mathbb{R}^{N \times N' \times K}$ is an indication tensor, where $\Omega^{(t+j)}_{o,d,k} = 1$ if the OD pair (o, d) is not empty in the $(t + j)$-th future interval. Note that although we aim to predict full tensors, the ground truth tensors are often still sparse, and thus we compute the errors taking into account only the non-empty elements in the ground truth tensors. Next, $\circ$ is element-wise multiplication, $\widehat{\mathbf{M}}^{(t+j)}$ and $\mathbf{M}^{(t+j)}$ are predicted and ground truth tensors, respectively, $||\cdot||_F$ is the Frobenius-norm.

V. Forecast with Spatial Dependency

To improve forecast accuracy, we propose an advanced framework that takes into account spatial dependency in *two* stages. First, in the factorization step, we use two graphs to model the spatial relationships among the origin and destination regions, respectively. Then, we apply graph convolutional neural networks to perform factorizations for origin and destination dimensions, respectively. Second, in the forecasting step, we integrate graph convolutions with RNNs to capture spatio-temporal correlations. The pseudo code of this framework is shown in Algorithm 2.

A. Spatial Factorization

As in Section IV-B, we aim to factorize tensor $\mathbf{M}^{(t-i+1)}$ into two smaller tensors $\mathbf{C}^{(t-i+1)}$ and $\mathbf{R}^{(t-i+1)}$. In the basic framework, $\mathbf{M}^{(t-i+1)}$ is simply flattened and followed by two fully-connected layers to construct $\mathbf{C}^{(t-i+1)}$ and $\mathbf{R}^{(t-i+1)}$. This process does not take spatial correlations among the origin regions and among the destination regions into account. However, such correlations very likely affect travel speeds. For example, if regions x and y are nearby, the travel speeds from regions x and y to the same destination region may tend to be correlated; similarly, the travel speeds from the same source

Algorithm 2 Advanced Forecasting Framework

Input:
Sparse OD stochastic speed tensors: $\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}$;
Output:
Full OD stochastic forecasts: $\widehat{\mathbf{M}}^{(t+1)}, \ldots, \widehat{\mathbf{M}}^{(t+h)}$;

Initialize CNRNN states in encoder: $\boldsymbol{H}_C^{(1)}, \cdots, \boldsymbol{H}_C^{(L)}$, and $\boldsymbol{H}_R^{(1)}, \cdots, \boldsymbol{H}_R^{(L)}$;
Initialize CNRNN states in decoder: $\hat{\boldsymbol{H}}_C^{(1)}, \cdots, \hat{\boldsymbol{H}}_C^{(L)}$, and $\hat{\boldsymbol{H}}_R^{(1)}, \cdots, \hat{\boldsymbol{H}}_R^{(L)}$;
for $i = t-s+1, \cdots, t$ **do**
  $\mathbf{R}^{(0)} \leftarrow \text{GCNN}_R(\mathbf{M}^{(i)})$, $\mathbf{C}^{(0)} \leftarrow \text{GCNN}_C(\mathbf{M}^{(i)})$; (Section V-A)
  for $l = 1, \cdots, L$ **do**
    $\mathbf{R}^{(l)} = \boldsymbol{H}_R^{(l)} \leftarrow \text{CNRNN}_{enc}^R(\mathbf{R}^{(l-1)}, \boldsymbol{H}_R^{(l)})$;
    $\mathbf{C}^{(l)} = \boldsymbol{H}_C^{(l)} \leftarrow \text{CNRNN}_{enc}^C(\mathbf{C}^{(l-1)}, \boldsymbol{H}_C^{(l)})$; (Section V-B)
  end for
end for
for $i = t+1, \cdots, t+h$ **do**
  $\widehat{\mathbf{R}}^{(0)} \leftarrow \boldsymbol{H}_R^{(L)}$, $\widehat{\mathbf{C}}^{(0)} \leftarrow \boldsymbol{H}_C^{(L)}$;
  for $l = 1, \cdots, L$ **do**
    $\widehat{\mathbf{R}}^{(l)} = \hat{\boldsymbol{H}}_R^{(l)} \leftarrow \text{CNRNN}_{dec}^R(\widehat{\mathbf{R}}^{(l-1)}, \hat{\boldsymbol{H}}_R^{(l)})$;
    $\widehat{\mathbf{C}}^{(l)} = \hat{\boldsymbol{H}}_C^{(l)} \leftarrow \text{CNRNN}_{dec}^C(\widehat{\mathbf{C}}^{(l-1)}, \hat{\boldsymbol{H}}_C^{(l)})$; (Section V-B)
  end for
  $\widetilde{\mathbf{M}}^{(i)} \leftarrow \widehat{\mathbf{R}}^{(L)} \times (\widehat{\mathbf{C}}^{(L)})$ (See Eq. 2);
  $\widehat{\mathbf{M}}^{(i)} \leftarrow \text{softmax}(\widetilde{\mathbf{M}}^{(i)})$ (See Eq. 3);
end for
if Is training. **then**
  Compute loss $L(\mathbf{M}, \widehat{\mathbf{M}})$ (See Eq. 11);
  Minimize the loss through back-propagation;
end if

region to regions x and y may also tend to be correlated. To take into account such spatial correlations, we first capture spatial correlations among origins and among destination regions using two separate graphs; then the captured spatial correlations are employed to conduct factorization with the help of graph convolutional neural networks (GCNNs).

1) Capturing Spatial Correlations: We leverage the notion of a proximity matrix [38] to capture spatial correlations. We proceed to present the idea using origin regions as an example, which also applies to destination regions in a similar manner.

Given $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N \times N' \times K}$, we have N origin regions. We build a graph to model the spatial relationships among the regions, where a vertex represents a region. Then, we utilize an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ to represent the graph: $\boldsymbol{A}_{u,v} = 1$ if regions $\mathbb{V}_u$ and $\mathbb{V}_v$ are spatially adjacent; otherwise, $\boldsymbol{A}_{u,v} = 0$.

We then construct a weighted proximity matrix $\boldsymbol{W}^{(\alpha,\sigma)} \in \mathbb{R}^{N \times N}$ from $\boldsymbol{A}$ that describes the proximity between regions $\mathbb{V}_u$ and $\mathbb{V}_v$ and is parameterized by hops α and standard deviation σ. Specifically, if $\mathbb{V}_v$ can be reached from $\mathbb{V}_u$ in

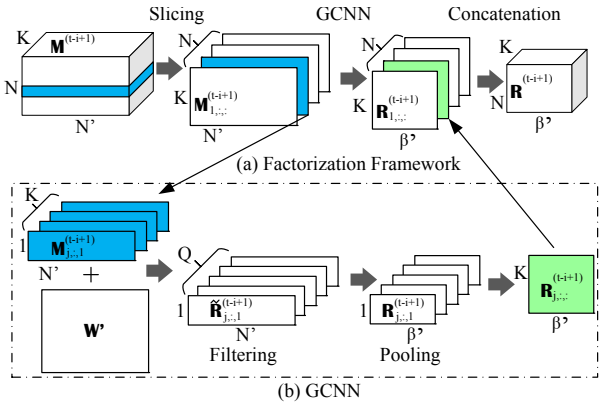


Figure 4: Spatial Factorization for $\boldsymbol{R}$

α hops using $\boldsymbol{A}$, $\boldsymbol{W}_{u,v}^{(\alpha,\sigma)} = e^{-x^2/\sigma^2}$, where x is the distance between the centroid of $\mathbb{V}_u$ and $\mathbb{V}_v$; otherwise $\boldsymbol{W}_{u,v}^{(\alpha,\sigma)} = 0$. In the experiments, we study the effect of α and σ (see Section VI-B4). The proximity matrix $\boldsymbol{W}^{(\alpha,\sigma)}$ is symmetric and non-negative.

The adjacency matrices for the source regions and destination regions may be different or the same. Consider two scenarios. First, we use OD matrices to model the travel costs within a city. In this case, the source regions and the destination regions are the same, and thus the two adjacency matrices are the same. Second, we may use OD matrices to model the travel costs between two different cities. Then, the source regions and the destination regions are in different cities. Thus, we need two different adjacency matrices. To avoid confusion, we use $\boldsymbol{W}$ and $\boldsymbol{W}'$ represent the proximity matrices for source regions and destination regions, respectively.

2) Factorization: We proceed to show the factorization procedure that utilizes the captured spatial correlations, i.e., $\boldsymbol{W}$ and $\boldsymbol{W}'$. Specifically, we show how to obtain $\mathbf{R}^{(t-i+1)}$ from $\mathbf{M}^{(t-i+1)}$. The same procedure can be applied to obtain $\mathbf{C}^{(t-i+1)}$.

The spatial factorization framework is shown in Figure 4(a). We first slice $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N \times N' \times K}$ by the origin region dimension into N matrices, i.e., $slice(\mathbf{M}^{(t-i+1)}) = [\mathbf{M}_{1,:,:}^{(t-i+1)}, \cdots, \mathbf{M}_{N,:,:}^{(t-i+1)}]$. Each of the sliced matrix represents to the costs from a specific origin region to all destination regions. To take into account the spatial correlations among destination regions, we apply a graph convolutional neural network (GCNN) operation using $\boldsymbol{W}'$, i.e., the proximity matrix for destination regions. The GCNN operation is able to take into account the spatial correlations among destination regions to generate hidden features such that nearby destination regions share similar hidden features in each $\mathbf{R}_{j,:,:}^{(t-i+1)}$, $j \in [1, N]$. The GCNN operation is applied on each slice and we obtain the GCNN output as $[\mathbf{R}_{1,:,:}^{(t-i+1)}, \cdots, \mathbf{R}_{N,:,:}^{(t-i+1)}]$. We then concatenate them to obtain $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N \times \beta' \times K}$.

We proceed to elaborate the specifics of applying graph convolution on a sliced matrix $\mathbf{M}_{j,:,:}^{(t-i+1)}$ $j \in [1, N]$ using Figure 4(b), which consists of *Filtering* and *Pooling*.

Filtering: Given $\mathbf{M}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{K \times N'}$, we apply Q graph

convolutional filters, which take into account the destination region adjacency matrix $\boldsymbol{W'}$, to generate $\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{N' \times Q}$ that captures the correlated features among destination regions.

We first slice $\mathbf{M}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{K \times N'}$ into K vectors $[\mathbf{M}_{j,:,1}^{(t-i+1)}, \cdots, \mathbf{M}_{j,:,K}^{(t-i+1)}]$, where vector $\mathbf{M}_{j,:,k}^{(t-i+1)} \in \mathbb{R}^{N'}$, $k \in [1, K]$, represents the probability of speeds falling into the k-th speed bucket when traveling from origin region $\mathbb{V}_j$ to all destination regions.

Next, we use a specific graph convolutional filter, namely Cheby-Net, due to its high accuracy and efficiency [32], on each vector $\mathbf{M}_{j,:,k}^{(t-i+1)}$. Specifically, before conducting actual convolutions, we compute $\boldsymbol{T}_k^{(t-i+1)} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_S]$, where $\boldsymbol{t}_s \in \mathbb{R}^{N'}$, $s \in [1, S]$, from $\mathbf{M}_{j,:,k}^{(t-i+1)}$. Here, $\boldsymbol{t}_1 = \mathbf{M}_{j,:,k}^{(t-i+1)}$, $\boldsymbol{t}_2 = \hat{\boldsymbol{L}} \times \mathbf{M}_{j,:,k}^{(t-i+1)}$, and $T_s = 2\hat{\boldsymbol{L}} \times T_{s-1} - T_{s-2}$ when $s > 2$, where $\hat{\boldsymbol{L}} = 2\boldsymbol{L}/\lambda_{max} - \boldsymbol{I}$ is a scaled Laplacian matrix and where $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W'}$ is the Laplacian matrix and λ_{max} is the maximum eigenvalue of $\boldsymbol{L}$. Here, we use destination adjacency matrix $\boldsymbol{W'}$ because $\mathbf{M}_{j,:,k}^{(t-i+1)}$ represents the speed from source region j to all destination regions and we use $\boldsymbol{W'}$ to capture the spatial correlation among destination regions. After the whole computation, we get $\boldsymbol{T}_k \in \mathbb{R}^{N' \times S}$ as the encoded features for the k-th bucket while considering the spatial correlations among destination regions.

Then we proceed to apply Q filters to $\boldsymbol{T}_k$. Each filter is a vector $\mathbf{G}_q \in \mathbb{R}^S$, where $q \in [1, Q]$. We apply each filter to all $\{\boldsymbol{T}_k^{(t-i+1)}\}$, $\forall k \in [1, K]$, and then the sum is used as the output of the filter.

$$\widetilde{\mathbf{R}}_{j,:,q}^{(t-i+1)} = \mathbf{G}_q \otimes \mathbf{M}_{j,:,:}^{(t-i+1)} = \sum_{k=1}^{K} \big(\epsilon(\boldsymbol{T}_k^{(t-i+1)} \times \mathbf{G}_q + \boldsymbol{b}_q), \quad (5)$$

where $\otimes$ denotes the Cheby-Net graph convolution operation, $\boldsymbol{b}_q \in \mathbb{R}^{N'}$ is a bias vector, and $\epsilon(\cdot)$ is a non-linear activate function.

Finally, we arrange the results obtained from all Q filters as $\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)} = [\widetilde{\mathbf{R}}_{j,:,1}^{(t-i+1)}, \ldots, \widetilde{\mathbf{R}}_{j,:,Q}^{(t-i+1)}]$, where $\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{Q \times N'}$.

Pooling: To further condense the features and to construct the final factorizations, we apply geometrical pooling [32] to $\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)}$ over the destination region dimension to obtain $\mathbf{R}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{Q \times \beta'}$, where $\beta' = \frac{N'}{p}$ and p are the pooling and stride size, respectively. This process is shown as follows.

$$\mathbf{R}_{j,:,:}^{(t-i+1)} = \mathrm{P}(\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)}), \quad (6)$$

where $\mathrm{P}(\cdot)$ is the pooling function that can be either max pooling or average pooling.

Since the pooling operation requires meaningful neighborhood relationships, we identify spatial clusters of destination regions. For example, in Figure 1(b), if we use the order of ascending region ids, i.e., (1, 2, 3, 4, 5, 6, 7, 8) to conduct pooling with a pooling size of 2, then regions 3 and 4 are pooled together. However, regions 3 and 4 are not neighbors, so this procedure may yield inferior features that may in turn yield undesired results. Instead, if we identify clusters of regions, we are able to produce a new order, e.g., (6, 1, 2, 3, 5, 4, 7, 8). When again using a pooling size of 2, each pool contains neighboring regions.

The GCNN process, including filtering and pooling, is repeated several times with different numbers of filters Q and pooling stride size p. Eventually, we set $Q = K$ and get $\mathbf{R}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{\beta' \times K}$.

As shown in Figure 4(a), the last operation is concatenation. We concatenate $\mathbf{R}_{j,:,:}^{(t-i+1)}$, $j \in [1, N]$ to obtain $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N \times \beta' \times K}$.

The same procedure can be applied to obtain $\mathbf{C}^{(t-i+1)}$ where we need to change $\boldsymbol{W'}$ to $\boldsymbol{W}$ when conducting the graph convolution.

B. Spatio-temporal Forecasting

To model temporal dynamics while keeping the spatial correlations in RNNs, we combine Cheby-Net based graph convolution with RNNs, yielding CNRNNs. Intuitively, we follow the structure of gated recurrent units while replacing the traditional fully connected layer by a Cheby-Net based graph convolution layer. Separate CNRNNs are employed to process $\mathbf{R}^{(t)}$ and $\mathbf{C}^{(t)}$.

Taking the source region dimension as an example, a CNRNN takes as input $\mathbf{R}^{(t)}$ at time interval $T^{(t)}$, and it predicts $\widehat{\mathbf{R}}^{(t+1)}$ for the future time interval $T^{(t+1)}$. This procedure is formulated as follows.

$$\mathbf{S}^{(t+1)} = \sigma(\mathbf{G_S} \otimes [\mathbf{H}^{(t)} : \mathbf{R}^{(t)}] + \boldsymbol{b_S}) \quad (7)$$

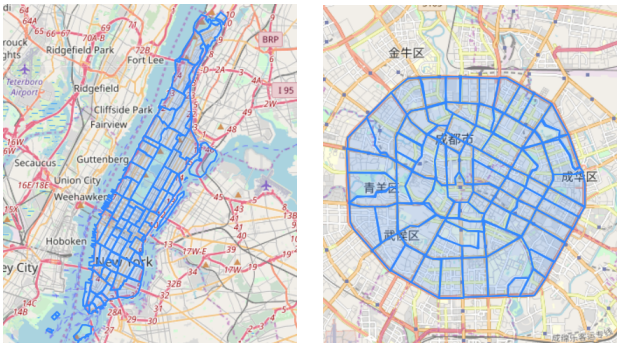
$$\mathbf{U}^{(t+1)} = \sigma(\mathbf{G_U} \otimes [\mathbf{H}^{(t)} : \mathbf{R}^{(t)}] + \boldsymbol{b_U}) \quad (8)$$

$$\mathbf{H}^{(t+1)} = \tanh(\mathbf{G_H} \otimes [\mathbf{R}^{(t)} : (\mathbf{S}^{(t+1)} \circ \mathbf{H}^{(t)})] + \boldsymbol{b_H}) \quad (9)$$

$$\widehat{\mathbf{R}}^{(t+1)} = \mathbf{U}^{(t+1)} \circ \mathbf{R}^{(t)} + (1 - \mathbf{U}^{(t+1)}) \circ \mathbf{H}^{(t+1)} \quad (10)$$

where $\mathbf{G_S}$, $\mathbf{G_U}$, and $\mathbf{G_H}$ are graph convolution filters; $\mathbf{R}^{(t)}$ and $\widehat{\mathbf{R}}^{(t+1)}$ are the input and output of a CNRNN cell at time interval $T^{(t)}$, respectively; $\mathbf{S}^{(t)}$ and $\mathbf{U}^{(t)}$ are the reset and update gates, respectively; $\otimes$ denotes the Cheby-Net graph convolution which defined in Equation 5, and here the graph convolution should take into account source adjacency matrix $\boldsymbol{W}$ since $\mathbf{R}$ captures features of source regions. $\circ$ denotes the Hadamard product between two tensors; and $\epsilon(\cdot)$, $\sigma(\cdot)$, and $\tanh(\cdot)$ are non-linear activation functions.

When applying CNRNN to predict $\widehat{\mathbf{C}}^{(t+1)}$, we need to change $\boldsymbol{W}$ to $\boldsymbol{W'}$ when conducting the graph convolution as $\mathbf{R}$ captures features of destination regions. Given predicted factorization tensors $[\widehat{\mathbf{R}}^{(t+1)}, \ldots, \widehat{\mathbf{R}}^{(t+h)}]$ and $[\widehat{\mathbf{C}}^{(t+1)}, \ldots, \widehat{\mathbf{C}}^{(t+h)}]$, we apply the same recovery operation introduced in Section IV-D to obtain h full OD stochastic speed tensors for the future time intervals $T^{(t+1)}, \ldots, T^{(t+h)}$ as the recovery output: $\widehat{\mathbf{M}}^{(t+1)}, \ldots, \widehat{\mathbf{M}}^{(t+h)}$.



(a) NYC Regions (b) CD Regions

Figure 5: Regions of NYC and CD

C. Loss Function

Similar to the construction covered in Section IV-E, we present the loss function as follows.

$$\ell = \sum_{j=1}^{h} [\lambda ||\widehat{\mathbf{R}}^{(t+j)}||^2_{\boldsymbol{W}} + \lambda ||\widehat{\mathbf{C}}^{(t+j)}||^2_{\boldsymbol{W}} + ||\Omega^{(t+j)} \circ (\mathbf{M}^{(t+j)} - \widehat{\mathbf{M}}^{(t+j)})||^2_F] \tag{11}$$

where $||\cdot||^2_{\boldsymbol{W}}$ is the Dirichlet norm under the proximity matrix $\boldsymbol{W}$, $\lambda_{\widehat{\mathbf{R}}}$ and $\lambda_{\widehat{\mathbf{C}}}$ are the regularization parameters for $\widehat{\mathbf{R}}$ and $\widehat{\mathbf{C}}$, respectively.. We use the Dirichlet norm because it takes the adjacency matrix into account—nearby regions should share similar features in the dense tensors $\mathbf{R}$ and $\mathbf{C}$. Finally, $\Omega^{(t+j)} \in \mathbb{R}^{N \times N' \times K}$ is an indication tensor (as defined in Equation 4), and $\widehat{\mathbf{M}}^{(t+j)}$ and $\mathbf{M}^{(t+j)}$, $j \in [1, h]$, are the predicted and ground truth tensors, respectively.

VI. Experiments

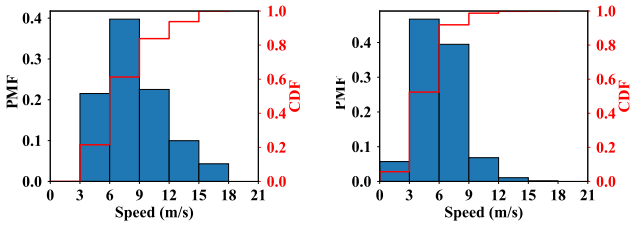
We describe the experimental setup and then present the experiments and the findings. The source code is available at https://github.com/hujilin1229/od-pred.

A. Experimental Setup

1) Datasets: **New York City Data Set (NYC):** We use 14 million taxi trips collected from 2013-11-01 to 2013-12-31 from Manhattan, New York City. Each trip consists of a pickup time, a drop off time, a pickup location, a drop off location, and a total distance. Manhattan has 67 taxizones[1], each of which is used as a region. We utlize the taxizone shapefile again to segment Manhattan borough into 67 subregions, as is shown in Figure 5(a).

For both data sets, we represent a stochastic speed (m/s) as a histogram with 7 buckets $[0, 3)$, $[3, 6)$, $[6, 9)$, $[9, 12)$, $[12, 15)$, $[15, 18)$, and $[18, \infty)$. Since 15 minutes is the most frequently used statistical interval in traffic studies [10], we consider 15-minutes intervals and obtain 96 intervals per day. The OD stochastic speeds during an interval for NYC are represented as an $\mathbb{R}^{67 \times 67 \times 7}$ tensor.

Chengdu Data Set (CD): CD contains 1.4 billion GPS records from 14,864 taxis collected from 2014-08-03 to 2014-08-30 in Chengdu, China[2]. Each GPS record consists of a taxi ID, a latitude, a longitude, an indicator of whether the taxi is occupied, and a timestamp. We consider sequences of GPS records where taxis were occupied as trips. We use a total of 3,636,845 trips that occurred within the second ring road of Chengdu. Next, we partition Chengdu within the second ring road into 79 regions according to the main roads, as shown in Figure 5(b)). The OD stochastic speeds during an interval for CD are represented as an $\mathbb{R}^{79 \times 79 \times 7}$ tensor.



(a) NYC Speed (b) CD Speed

Figure 6: Speed Distributions for NYC and CD

Figures 6(a) and 6(b) show the speed distributions for both datasets. Most speeds fall into the range from 0 to 21m/s. Thus, with the selected histogram buckets, the generated distributions are meaningful.

We say that an OD pair is valid if it has at least one speed record. Figures 7(a) and 7(b) show the averaged percentage of valid OD pairs in different time intervals. In Figure 7(a), we see that fewer than 5% of all OD pairs on average are covered with data at any time interval in NYC. The sparseness is even more severe in CD, where fewer than 1% of all OD pairs on average are covered. This severe data sparseness, renders it difficult to evaluate the models properly. We say that a data record is unqualified if the number of valid OD pairs is less than a threshold. Therefore, we pre-process both data sets, removing unqualified data records, that are too sparse. We adopt a thresholds of 100 and 40 number of valid OD pairs for NYC and CD, respectively. Specifically, we select the time intervals with at least 100 OD pairs contain travel records in NYC, and 40 OD pairs in CD. Figure 7(b) shows the sparseness after preprocessing, where the data sparseness in our finalized data is hugely reduced. The thresholds chosen represent is a trade-off between the amount of data available for evaluation and the sparseness of the data used for evaluation.

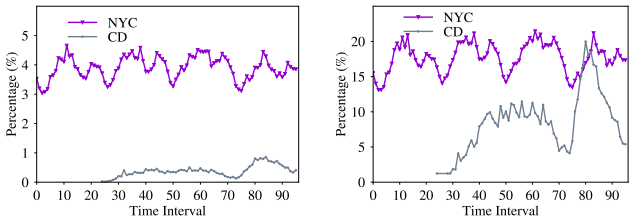
After preprocessing, we select the first 70% of the data as training data, the next 10% as validation data, and the remaining 20% as testing data. We apply the same selection procedure to both NYC and CD. Further, we split the training data into non-overlapping batches of length $s + h$.

2) Forecast Settings: We consider settings where $s = 3$ or $s = 6$ while varying h among 1, 2, and 3. This means that we use stochastic OD matrices from 3 or 6 historical intervals to predict stochastic OD matrices during up to 3 future intervals.

3) Baselines: To evaluate the effectiveness of the proposed base framework (BF) and advanced framework (AF), we compare with 5 different methods. Since only OD pairs are given,

[1] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[2] https://goo.gl/3VsEym



(a) Original Data (b) Preprocessed Data

Figure 7: Spareness of Original and Preprocessed Data

not specific segments or paths, we do not consider segment-based and path-based methods, which cannot be applied directly. Instead, we extend two state-of-the-art OD-pair based methods to support stochastic weights: (1) recurrent neural networks using GRU gates (RNN) [30] and (2) Multi-task Representation Learning (MR) [2]. In addition, we consider the following baselines. (3) Naive Histograms (NH): for each OD pair, we use all travel speed records for the OD pair in the training data set to construct a histogram and use the histogram for predicting the future stochastic speeds. Next, we model the stochastic speeds for each OD pair as a time series of vectors, where each vector represents the stochastic speed of the OD pair in an interval. Based on this time series modeling, we consider two different time series forecasting methods: (4) Gaussian Process Regression (GP) [39], which considers the vectors independently; (5) Multi-variate vector autoregression (VAR) [40], which takes into account the linear correlations among different OD pairs. We do not compare with segment-based or path-based methods (cf. Section II) because the input to such methods is a path. In our setting, multiple paths exist from a source region to a destination region, thus rendering such methods inapplicable.

4) Evaluation Metrics: To quantify the effectiveness of the proposed frameworks, we use three commonly adopted distance functions for distributions, i.e., Kullback-Leibler divergence (KL), Jensen-Shannon divergence (JS), and earth-mover's distance (EMD), to measure the accuracy of forecasts.

All three functions capture the dissimilarity between an estimated distribution and a ground-truth distribution. Thus, low values are preferred. Specifically, the general dissimilarity metric is defined as follows.

$$\text{DisSim}_{\text{f}}^{(k)} = \frac{\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N'}\Omega_{i,j}^{(t+k)}\text{f}(\mathbf{M}_{i,j,:}^{(t+k)}, \widehat{\mathbf{M}}_{i,j,:}^{(t+k)})}{\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\Omega_{i,j}^{(t+k)}}, \tag{12}$$

where $1 \leq k \leq h$ denotes the k-th step ahead forecasts, T is the size of testing set. Next, $\Omega^{(t+k)}$, $\mathbf{M}^{(t+k)}$, and $\widehat{\mathbf{M}}^{(t+k)}$ are the indication matrix, ground truth tensor, and forcast tensor, respectively; $\Omega_{(i,j)}^{(t+k)} = 1$ if observations exist from region i to region j at step $(t+k)$; otherwise, $\Omega_{(i,j)}^{(t+k)} = 0$. Moreover, $\text{f}(\cdot)$ is a generic metric function that can be any of the metrics mentioned above and defined next. For simplicity, we use $\boldsymbol{m}$ and $\widehat{\boldsymbol{m}} \in \mathbb{R}^K$ to denote $\mathbf{M}_{i,j,:}^{(t+k)}$ and $\widehat{\mathbf{M}}_{i,j,:}^{(t+k)}$, respectively.

Data	Model	Configuration	#Weights
NYC	FC	FC_3-GRU_3^1-$FC_{31,423}$	408,535
	BF	$2\times FC_2$-GRU_2^1-$FC_{2,345}$	391,182
	BF	$2 \times GC_8^{32}$-$P4$-GC_4^{32}-$P2$-$GCR_2^{32\times4}$	339,726
CD	FC	FC_3-GRU_3^1-FC_{43687}	567,967
	BF	$2\times FC_2$-GRU_2^1-$FC_{2,765}$	415,339
	AF	$2 \times GC_8^{32}$-$P4$-GC_4^{32}-$P2$-$GCR_2^{32\times4}$	367,502

Table I: Model Construction and Hyper-Parameter Selection

KL divergence is defined as follows,

$$\text{KL}(\boldsymbol{m}, \widehat{\boldsymbol{m}}) = \sum_{k=1}^{K}\widehat{\boldsymbol{m}}_k \log(\frac{\widehat{\boldsymbol{m}}_k + \delta}{\boldsymbol{m}_k + \delta}), \tag{13}$$

where δ is a positive small value to prevent having a zero when using the log function. We use $\delta = 0.001$ in the experiment.

Jensen-Shannon divergence is defined as follows,

$$\text{JS}(\boldsymbol{m}, \widehat{\boldsymbol{m}}) = \frac{\text{KL}(\boldsymbol{m}, \overline{\boldsymbol{m}}) + \text{KL}(\widehat{\boldsymbol{m}}, \overline{\boldsymbol{m}})}{2}, \tag{14}$$

where $\overline{\boldsymbol{m}} = 0.5 \times (\boldsymbol{m} + \widehat{\boldsymbol{m}})$.

Earth mover's distance is defined as follows,

$$\text{EMD}(\boldsymbol{m}, \widehat{\boldsymbol{m}}) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{K}\boldsymbol{F}_{i,j}d_{i,j}}{\sum_{i=1}^{K}\sum_{j=1}^{K}\boldsymbol{F}_{i,j}}, \tag{15}$$

where flow matrix $\boldsymbol{F}$ is the optimal flow that minimizes the overal cost from $\boldsymbol{m}$ to $\widehat{\boldsymbol{m}}$ [41].

5) Model Construction: The proposed frameworks are trained by minimizing the two loss functions, defined in Equation 4 for BF and Equation 11 for AF, using back-propagation. We use the Adam optimizer due to its good performance. The hyper-parameters were configured manually based on the loss on the validation set. Specifically, we set the initialization learning rate to 0.001, the decay rate to 0.8 at every 5 epochs, and the dropout rate to 0.2. The chosen hyper-parameters' settings are given in Table I.

Table I shows the optimal configurations of the hyper-parameters for the three deep learning methods and numbers of weight parameters used in each model for both datasets. Baseline FC first encodes the input into a 2D latent space via an FC operation, denoted as FC_2. Then it calls a GRU with 3 units and 1 layer to capture the temporal dynamics, denoted as GRU_3^1. Finally, another FC is called to project the output from GRU to an OD stochastic tensor with the following dimensions: #Source Regions × #Destination Regions × #Buckets, e.g., $67 \times 67 \times 7 = 31,423$ dimensions for NYC, denoted as $FC_{31,423}$. For BF and AF, we apply two identical configurations for origin and destination factorization, which is why we have "2×" on the first configuration. For BF, we first utilize FC_2 to encode the input for the first factorization. Then we adopt GRU_2^1 to learn the temporal dynamics. At the end of GRU, we project the output into a corresponding factorization with the following dimensions: #Source Regions × r × #Buckets, where r is the rank of the factorized dense matrix, which we set to 5, e.g., $67\times5\times7 = 2,345$ for NYC, denoted as $FC_{2,345}$. The configuration for AF is very different from those of the previous two models. First, we adopt two combinations

of GCNN, GC_K^Q, where Q is the filter number and K is the filter size, and a pooling operation Pp, where p is the pooling size, e.g., GC_8^{32}-$P4$-GC_4^{32}-$P2$ for NYC. Then, the encoded features are fed into a CNRNN with n layers, where each layer has four Cheby-Nets. Assuming that the number and size of the filters are Q_c and K_c, this operation can be written as $GCR_n^{Q_c \times K_c}$, e.g., $GCR_2^{32\times 4}$, implying 2 CNRNNs where the GCNN in each gate has 32 graph convolutional filters of size 4. From the above configurations, although AF uses the most complex models, AF uses the fewest weight parameters (see the # weights column in Table I).

B. Experimental Results

1) Overall Results: We compare the accuracies of the different methods, using KL, JS, and EMD to evaluate the forecast accuracy. In Table II, we can first fix s at 3 or 6, while varying h, i.e., the h-intervals ahead forecasting. We also vary s, i.e., the number of historical stochastic speed matrices, to study the effect of s and h.

We have the following observations. (1) The deep learning based methods perform better than the other baselines in most cases. (2) The proposed basic framework BF performs better than other methods in most settings. This indicates that the proposed frameworks, which involve factorization and RNN based forecasting, are effective for OD matrix forecasting in settings with data sparseness. (3) The advanced framework AF is significantly better than other methods, including BF, in all settings. This suggests that by taking into account the spatial correlations among regions using two GCNNs, the learned features become more meaningful, which then improves forecasting accuracy. (4) The reuslts on NYC are better than those on CD. This is because the regions in NYC are more homogeneous (i.e., within Manhatten) than the regions in CD that cover a much larger and more diverse region. This in turn makes the traffic situations in CD much more complex and more challenging to forecast. (5) When varying h, the accuracy of AF becomes worse, i.e., larger metric values. This suggests that forecast far into the future becomes more challenge. (6) When fixing h, we compare the two tables and observe that the performance of AF is better at $s = 3$ than $s = 6$. This seems to indicate that the traffic variations are more dependent on short-term history (i.e., $s = 3$) than on long-term history (i.e., $s = 6$).

According to the above results, in the following, we only consider FC, BF, and AF, and we only consider the setting where $h = 1$ and $s = 6$, i.e., 1-step ahead forecasting with 6 historical observations.

2) Effect of Time of Day: In this experiment, we aim at investigating forecasting performance for different intervals during a day. To this end, we show the forecast accuracy across different time intervals. Figures 8, 9, and 10 show the performance on both data sets when using EMD, KL, and JS. To visualize the results across time, we aggregate the results per each 3 hours. We use three curves to represent the accuracy of FC, BF, and AF. In addition, we use bars to represent the percentages of data we have per each 3 hours. CD does not contain any data from 00:00 to 06:00, which is why the figures for CD start at 6.

Figures 8(a) and 8(b) show the accuracy based on EMD. We observe that both AF and BF outperform FC in almost all the time intervals. This suggests the effectiveness of factorization in the proposed framework when contending with data sparseness. In addition, AF has the best performance and differs clearly from FC and BF. This suggests that by further capturing spatial and spatio-temporal correlations improves the forecast accuracy.

We observe that the EMD for all the three methods is the worst in NYC during [3:00, 6:00). This is because the amount of testing data during [3:00, 6:00) is quite small, only accounting for around 1% of the total testing data. On both data sets, the best EMD values appear during [12:00, 15:00). This indicates that the traffic conditions during this time period seems to have the least dynamics thus making the forecasting less challenging. Similar trends can be observed when using KL and JS, as shown in Figures 9 and 10. Overall, the advanced framework AF achieves consistently the best forecasting performance on both datasets and on the three different evaluation metrics. More data enables more accurate forecasting.

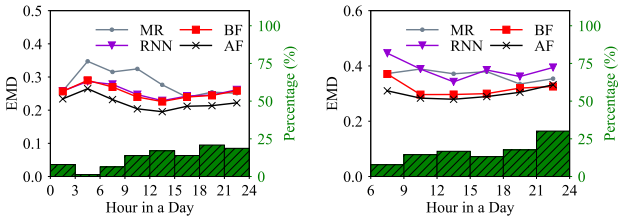
3) Effect of Distances: In this experiment, we aim at investigating the effect of the distances between source and destination regions. We thus report the forecast accuracy with different distances. Given a source and a destination region, we use the Euclidean distance between the centroids of the two regions as its corresponding distance. We group OD region pairs based on their distances into 6 groups as shown in Figures 11, 12, and 13. We only consider OD region pairs that are below 3 km because less than 1% of the data is available for OD region pairs more than 3 km apart. Figures 11, 12, and 13 report results on EMD, KL, and JS, respectively.

Figures 11(a) and 11(b) show the EMD values at varying distances on NYC and CD, respectively. We observe that (1) BF and AF outperform FC for all distance settings and on both datasets; (2) AF outperforms BF by a clear margin. This again offers evidence of effectiveness of the proposed advanced framework and suggests that the best performance is achieved by contending the sparseness and by capturing spatio-temporal correlations. Next, when considering distances from 0.5 to 1.5, i.e., the first three points of the curves, we observe a clear descending trend in NYC, but this trend is less obvious in CD.

We also observe that curves start to increase 1 km on NYC as shown in Figure 11(a). The reason is amount of data in distance range [1.5, 3.0] decreases quickly, in turn introducing more fluctuations. We observe a subtle trend from distance range [1., 1.5) to distance range [1.5, 2.0) in Figure 11(b) for BF and AF, however, FC is much worse in distance range [1.5, 3.0]. We have similar observations for NYC regarding the KL and JS evaluation metrics, which is shown in Figures 12(a) and 13(a). The trend is much clearer in CD for the evaluation metrics of KL and JS as is shown in Figures 12(b) and 13(b). Therefore, another explanation of the increasing tread is that

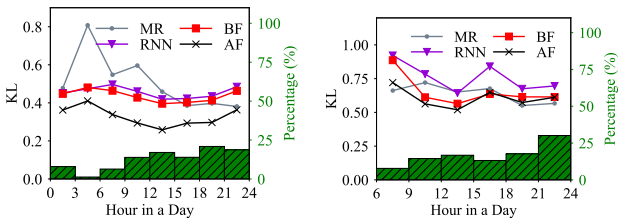
Data	Metric	h	s=6							s=3						
			NH	VAR	GP	MR	RNN	BF	AF	NH	VAR	GP	MR	RNN	BF	AF
NYC	KL	1	0.592	0.604	0.556	0.456	0.453	0.437	**0.313**	0.592	0.545	0.569	0.456	0.446	0.427	**0.311**
		2	0.592	0.659	0.586	0.456	0.445	0.421	**0.315**	0.592	0.566	0.575	0.456	0.438	0.417	**0.313**
		3	0.593	0.655	0.585	0.456	0.440	0.410	**0.317**	0.592	0.569	0.584	0.456	0.438	0.415	**0.314**
	JS	1	0.074	0.079	0.139	0.100	0.059	0.057	**0.052**	0.074	0.071	0.103	0.100	0.058	0.057	**0.054**
		2	0.074	0.085	0.139	0.100	0.058	0.056	**0.052**	0.074	0.073	0.104	0.100	0.057	0.056	**0.054**
		3	0.074	0.084	0.139	0.100	0.058	0.056	**0.053**	0.074	0.072	0.105	0.100	0.057	0.056	**0.055**
	EMD	1	0.293	0.310	1.794	0.271	0.248	0.238	**0.214**	0.293	0.284	1.116	0.271	0.250	0.246	**0.214**
		2	0.293	0.321	1.704	0.271	0.246	0.235	**0.216**	0.293	0.283	1.101	0.271	0.247	0.243	**0.216**
		3	0.294	0.329	1.701	0.271	0.245	0.233	**0.217**	0.294	0.281	1.082	0.271	0.247	0.243	**0.217**
CD	KL	1	0.686	0.792	0.671	0.624	0.753	0.658	**0.600**	0.697	0.731	0.674	0.624	0.694	0.582	**0.549**
		2	0.685	0.775	0.671	0.624	0.740	0.644	**0.605**	0.709	0.710	0.687	0.624	0.689	0.586	**0.555**
		3	0.686	0.808	0.674	0.624	0.735	0.637	**0.609**	0.700	0.694	0.684	0.624	0.807	0.792	**0.626**
	JS	1	0.090	0.107	0.090	0.131	0.091	0.079	**0.075**	0.091	0.097	0.091	0.131	0.085	0.073	**0.071**
		2	0.090	0.105	0.090	0.131	0.090	0.078	**0.076**	0.092	0.094	0.092	0.131	0.085	0.074	**0.072**
		3	0.090	0.112	0.090	0.131	0.089	0.078	**0.077**	0.092	0.092	0.093	0.131	0.097	0.092	**0.079**
	EMD	1	0.449	0.494	0.449	0.364	0.394	0.321	**0.304**	0.441	0.465	0.439	0.364	0.360	0.307	**0.289**
		2	0.448	0.488	0.448	0.364	0.389	0.319	**0.305**	0.443	0.444	0.434	0.364	0.361	0.313	**0.295**
		3	0.448	0.637	0.448	0.364	0.386	0.319	**0.306**	0.464	0.540	0.471	0.364	0.423	0.378	**0.311**

Table II: Forecast Accuracy with Varying h, $s = 6$ and $s = 3$



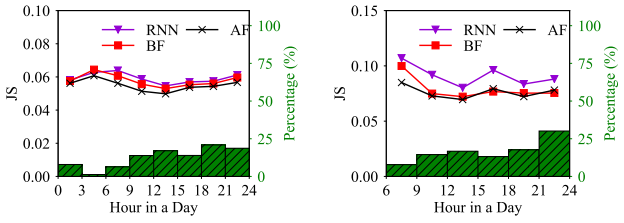
(a) NYC (b) CD

Figure 8: Effect of Time of a Day, EMD



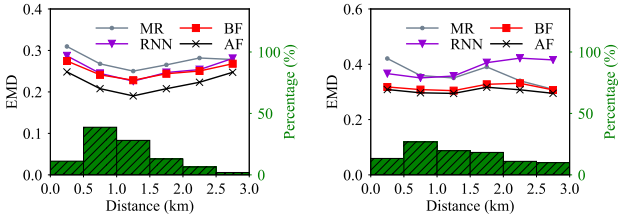
(a) NYC (b) CD

Figure 9: Effect of Time of a Day, KL



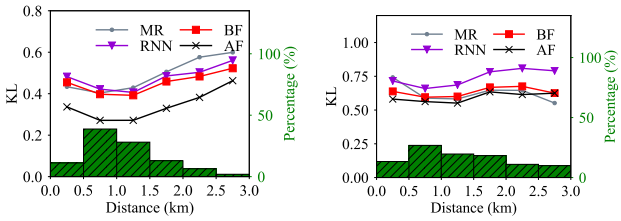
(a) NYC (b) CD

Figure 10: Effect of Time of a Day, JS



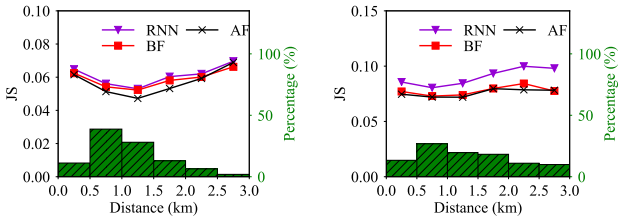
(a) NYC (b) CD

Figure 11: Effect of Distances, EMD



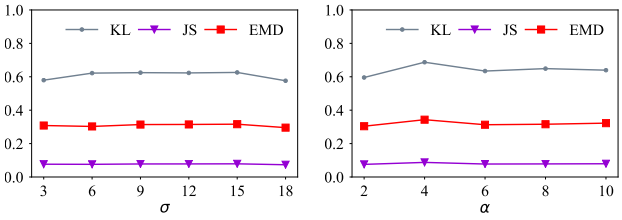
(a) NYC (b) CD

Figure 12: Effect of Distances, KL



(a) NYC (b) CD

Figure 13: Effect of Distances, JS



(a) Impact of σ (b) Impact of α

Figure 14: Results w.r.t. Hop α

as the distance increases, the route options also increase, which makes the speed more stochastic and harder to forecast. Overall, AF achieves the best performance on both datasets regarding to EMD, KL, and JS evaluation metrics.

4) Effect of Proximity Matrices: We conduct a last set of experiments to investigate the effect of the parameters σ and α when constructing the proximity matrix $\boldsymbol{W}$ in the advanced framework. We only report results for CD due to the space limitation and because NYC yields similar results. Figures 14(a) and 14(b) show the accuracy when varying α

and σ. The proposed AF is insensitive to σ and α. In other words, using proximity matrices is a robust way of capturing spatial correlations.

VII. Conclusions and Outlook

An increasingly pertinent settings are asking for full OD matrices contain stochastic travel costs between any pair of regions in near future. However, instantiating such OD matrices calls for a large amount of vehicle trajectories which is almost an impossible task to fulfill in reality. We define and study the problem of stochastic origin-destination matrix forecasting in this setting, which solves data sparseness and spatio-temporal correlations. Empirical studies on two real datasets from different countries, New York City and Chengdu City, demonstrate that the proposed framework outperforms other methods in all the experimental settings.

In future work, it is of interest to extend the framework to incorporate contextual information such as weather conditions and to consider the information at different timestamps differently, e.g., using attention networks [27]. Further, it is of interest to be able to avoid outlier predictions [42], [43].

Acknowledgments

This research was supported in part by a grant from Independent Research Fund Denmark, and by the DiCyPS center, funded by Innovation Fund Denmark.

References

[1] J. Hu, B. Yang, C. S. Jensen, and Y. Ma, "Enabling time-dependent uncertain eco-weights for road networks," *GeoInformatica*, vol. 21, no. 1, pp. 57–88, 2017.

[2] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *KDD*, pp. 1695–1704, 2018.

[3] S. A. Pedersen, B. Yang, and C. S. Jensen, "Fast stochastic routing under time-varying uncertainty.," *VLDB J.*, to appear, 2020.

[4] J. Hu, B. Yang, C. Guo, and C. S. Jensen, "Risk-aware path selection with time-varying, uncertain travel costs: a time series approach," *VLDB J.*, vol. 27, no. 2, pp. 179–200, 2018.

[5] H. Liu, C. Jin, B. Yang, and A. Zhou, "Finding top-k shortest paths with diversity," in *ICDE*, pp. 1761–1762, 2018.

[6] H. Wang, Y.-H. Kuo, D. Kifer, and Z. Li, "A simple baseline for travel time estimation using large-scale trip data," GIS, pp. 61:1–61:4, 2016.

[7] C. Guo, B. Yang, J. Hu, and C. S. Jensen, "Learning to route with sparse trajectory sets," in *ICDE*, pp. 1073–1084, 2018.

[8] J. Rice and E. Van Zwet, "A simple and effective method for predicting travel times on freeways," in *IEEE ITSC*, pp. 227–232, 2001.

[9] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *KDD*, pp. 316–324, 2011.

[10] B. Yang, C. Guo, and C. S. Jensen, "Travel cost inference from sparse, spatio-temporally correlated time series using markov models," *PVLDB*, vol. 6, no. 9, pp. 769–780, 2013.

[11] C. Wu, J. Ho, and D. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276–281, 2004.

[12] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *ICDE*, pp. 136–147, 2014.

[13] B. Yang, C. Guo, Y. Ma, and C. S. Jensen, "Toward personalized, context-aware routing," *VLDB J.*, vol. 24, no. 2, pp. 297–318, 2015.

[14] S. A. Pedersen, B. Yang, and C. S. Jensen, "A hybrid learning approach to stochastic routing," in *ICDE*, to appear, 2020.

[15] J. Hu, C. Guo, B. Yang, and C. S. Jensen, "Stochastic weight completion for road networks using graph convolutional networks," in *ICDE*, pp. 1274–1285, 2019.

[16] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu, "PACE: a path-centric paradigm for stochastic path finding," *VLDB J.*, vol. 27, no. 2, pp. 153–178, 2018.

[17] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu, "Path cost distribution estimation using trajectory data," *PVLDB*, vol. 10, no. 3, pp. 85–96, 2016.

[18] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *KDD*, pp. 25–34, 2014.

[19] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *KDD*, pp. 858–866, 2018.

[20] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? Estimating travel time based on deep neural networks," in *AAAI*, pp. 2500–2507, 2018.

[21] H. Zhang, H. Wu, W. Sun, and B. Zheng, "Deeptravel: a neural network based travel time estimation model with auxiliary supervision," *arXiv preprint arXiv:1802.02147*, 2018.

[22] T. Hunter, R. Herring, P. Abbeel, and A. Bayen, "Path and travel time inference from GPS probe vehicle data," *Analyzing Networks and Learning with Graphs Workshop held in NIPS*, vol. 12, no. 1, p. 2, 2009.

[23] S. B. Yang and B. Yang, "Learning to rank paths in spatial networks," in *ICDE*, to appear, 2020.

[24] I. Jindal, Tony, Qin, X. Chen, M. Nokleby, and J. Ye, "A unified neural network approach for estimating travel time and distance for a taxi trip," *ArXiv e-prints*, Oct. 2017.

[25] X. Dai, L. Sun, and Y. Xu, "Short-term origin-destination based metro flow prediction with probabilistic model selection approach," *Journal of Advanced Transportation*, vol. 2018, 2018.

[26] R. Cirstea, D. Micu, G. Muresan, C. Guo, and B. Yang, "Correlated time series forecasting using multi-task deep neural networks," in *CIKM*, pp. 1527–1530, 2018.

[27] R.-G. Cirstea, B. Yang, and C. Guo, "Graph attention recurrent neural networks for correlated time series forecasting.," in *MileTS19@KDD*, 2019.

[28] B. Zheng, K. Zheng, X. Xiao, H. Su, H. Yin, X. Zhou, and G. Li, "Keyword-aware continuous knn query on road networks," in *ICDE*, pp. 871–882, 2016.

[29] B. Zheng, H. Su, W. Hua, K. Zheng, X. Zhou, and G. Li, "Efficient clue-based route search on road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 9, pp. 1846–1859, 2017.

[30] F. Toque, E. Côme, M. K. E. Mahrsi, and L. Oukhellou, "Forecasting dynamic public transport origin-destination matrices with long-short term memory recurrent neural networks," in *ITSC*, pp. 1071–1076, 2016.

[31] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Distinguishing trajectories from different drivers using incompletely labeled trajectories," in *CIKM*, pp. 863–872, 2018.

[32] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," *arXiv:1606.09375 [cs, stat]*, June 2016.

[33] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv:1312.6203 [cs]*, 2013.

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

[35] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *NIPS*, pp. 3700–3710, 2017.

[36] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, pp. 3104–3112, 2014.

[37] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[38] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *ICLR*, 2018.

[39] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*, pp. 63–71, Springer, 2004.

[40] C. A. Sims, "Macroeconomics and reality," *Econometrica: Journal of the Econometric Society*, pp. 1–48, 1980.

[41] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *ICCV*, pp. 59–66, 1998.

[42] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in *IJCAI*, pp. 2725–2732, 2019.

[43] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *MDM*, pp. 125–134, 2018.