PeriodicMove: Shift-aware Human Mobility Recovery with Graph Neural Network

Hao Sun*
sunhao9908@gmail.com
University of Electronic Science and Technology of China
Chengdu, China

Changjie Yang*
yangcj12138@gmail.com
University of Electronic Science and Technology of China
Chengdu, China

Liwei Deng*
denglw0830@gmail.com
University of Electronic Science and Technology of China
Chengdu, China

Fan Zhou
zhoufan31@gmail.com
University of Electronic Science and Technology of China
Chengdu, China

Feiteng Huang
huangfeiteng@huawei.com
Huawei Cloud Database Innovation Lab
Chengdu, China

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

ABSTRACT

Human mobility recovery is of great importance for a wide range of location-based services. However, recovering human mobility is not trivial because of three challenges: 1) complex transition patterns among locations; 2) multi-level periodicity and shifting periodicity of human mobility; 3) sparsity of the collected trajectory data. In this paper, we propose PeriodicMove, a neural attention model based on graph neural network for human mobility recovery from lengthy and sparse trajectories. In PeriodicMove, we first construct a directed graph for each trajectory and capture complex location transition patterns using graph neural network. Then, we design two attention mechanisms which capture multi-level periodicity and shifting periodicity of human mobility respectively. Finally, a spatial-aware loss function is proposed to incorporate spatial proximity into the model optimization, which alleviates the data sparsity problem. We perform extensive experiments and the evaluation results demonstrate that PeriodicMove yields significant improvements over the competitors on two representative real-life mobility datasets. In addition, by providing high-quality mobility data, our model can benefit a variety of mobility-oriented downstream applications.

CCS CONCEPTS

• **Information systems → Location based services**; • **Human-centered computing → Ubiquitous and mobile computing design and evaluation methods**.

*These authors contributed equally to this research.
†Corresponding author.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482284

KEYWORDS

human mobility; trajectory recovery; graph neural network; attention mechanism

ACM Reference Format:
Hao Sun, Changjie Yang, Liwei Deng, Fan Zhou, Feiteng Huang, and Kai Zheng. 2021. PeriodicMove: Shift-aware Human Mobility Recovery with Graph Neural Network. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482284

1 INTRODUCTION

The widespread adoption of location-based services have accumulated large-scale human mobility data, which is of great importance in a broad spectrum of applications, ranging from personalized location recommendation [6, 41, 51] to urban transportation planning [47]. However, data quality issues (e.g., geolocation information missing, unreal check-ins, data sparsity) in real-life mobility data harm the performance of downstream applications. For example, with a limited number of trajectory records per day, it is difficult for recommender system to recommend a proper point of interest [39]. As for urban transportation planning, the missing data problem remains a challenge because most existing analyzing methods rely on complete data [15]. Therefore, it has become a pressing need to recover individual trajectories by understanding the mobility pattern from existing human mobility data.

A common solution to the task of human mobility recovery is to impute the missing location by treating individual trajectories as two-dimensional time series with latitude and longitude at each timestamp. Smoothing filters [1, 25] and LSTM-based methods [2, 34] have been proposed with acceptable performances when only a small percentage of locations are missing. However, in highly sparse scenarios, their performances deteriorate significantly because they fail to effectively model complex human mobility patterns. Another line of study focuses on adopting mobility prediction methods [30, 44] for human mobility recovery. However, these methods only consider the locations visited before and ignore the locations visited after the missing time slot. Recently, more researchers turn to model-based methods [39, 40]. They leverage attention mechanism and

sequential statistical models (e.g., recurrent neural network) to capture the mobility regularity of users and model spatial-temporal dependency among different locations so as to generate the missing locations.

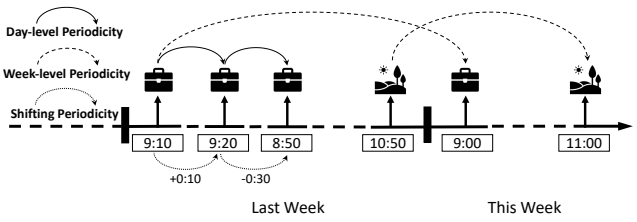


Figure 1: A running example of multi-level and shifting periodicity of human mobility.

Despite the inspiring results of the model-based methods mentioned above, human mobility recovery is still challenging for the following reasons: (1) Firstly, previous work [9] reveals that human mobility exhibits complex location transition patterns, but these methods always model simple transitions between consecutive locations and neglect more complex transitions in broader contexts, i.e., other locations in the trajectory. Thus, complex transitions among distant locations are often overlooked by these methods. (2) Secondly, human mobility periodicity is often hard to capture due to its multi-level and shifting nature. Consider the following running example shown in Figure 1, during weekdays, a person goes to work every morning, which shows the day-level periodicity of human mobility. However, when it comes to weekend, he would go to park for fun, which shows the week-level periodicity of human mobility. In fact, human mobility's periodicity of daily routines, weekly leisure and yearly festivals can be completely different, which demonstrates the multi-level nature of human mobility periodicity. Besides, the specific time this person arrives at workplace shifts from 9:10 to 9:20 and from 9:20 to 8:50, which shows the shifting nature of human mobility periodicity caused by environmental factors (e.g., traffic jams). Thus, a model which can capture the two natures of human mobility periodicity simultaneously is needed. (3) Thirdly, human mobility data is sparse because most data recording human mobility is low-sampling in nature, and the location information is recorded only when the user accesses the location service. Such sparsity hinders us from capturing spatial relations among locations, which is an important factor since it is impossible for people to go far away within a short period of time.

To deal with the challenges mentioned above, we propose PeriodicMove, a novel model to recover human mobility from lengthy and sparse individual trajectories. For clarity, we define the target day's trajectory as the current trajectory and any trajectory before the target day as historical trajectories. In PeriodicMove, we firstly construct a directed graph for each trajectory. Based on the graph, GNN is capable of capturing complex location transition patterns and generate location embedding correspondingly, followed by a temporal embedding layer to integrate spatial-temporal dependency. To capture the shifting periodicity of human mobility, we design a cross attention layer to process each historical trajectory separately and distill shifting periodicity from them. Specifically, cross attention mechanism yields attention weights to select locations from all the time slots of each historical trajectory based on current mobility status, and thus incorporate shifting periodicity information into the current time slot of each historical trajectory. To capture the multi-level periodicity of human mobility in a principled way, soft attention layer is trained to select the historical trajectories that are highly related to the current trajectory. Finally, a trajectory recovery layer is designed to fuse the features from historical trajectories and current trajectory, and generate the missing locations. Better still, we design a spatial-aware loss function to incorporate spatial proximity information into the model and alleviate the data sparsity problem.

Our contributions can be summarized as follows:

- We propose a novel neural attention model based on graph neural network called PeriodicMove, to recover human mobility from lengthy and sparse individual trajectories. Our model captures complex location transition patterns, multi-level periodicity and shifting periodicity of human mobility in a principled way. To the best of our knowledge, PeriodicMove is the first model that simultaneous combines these important features for accurate human mobility recovery.
- We leverage graph neural network for complex location transition patterns modeling. We design two attention mechanisms that are tailored to capture the multi-level periodicity and shifting periodicity of human mobility respectively. We propose a spatial-aware loss function to incorporate spatial proximity into the model, which alleviates the data sparsity problem.
- We perform extensive experiments on two representative real-life mobility datasets. Results show that PeriodicMove outperforms state-of-the-art models in terms of improving recovery accuracy by 2.9%~9.0%, which is quite remarkable due to the challenging nature of this task.

2 RELATED WORK

2.1 Human Mobility Recovery

Human mobility recovery is an important problem for it can benefit a wide range of downstream tasks. Existing works can be divided into three lines. The first line is to treat human mobility recovery problem as time series data recovery problem [21], which has been studied extensively. However, these models' performances are not acceptable in highly sparse scenarios due to its inability to capture human mobility patterns. The second line focuses on adopting mobility prediction models for human mobility recovery [9]. Their performances are acceptable when only a small percentage of locations are missing. However, their performances also decline in highly sparse scenarios because they only leverage the historical information before the missing location and fail to model the spatial-temporal dependency between the missing location and the locations visited afterwards. As such, models for mobility data recovery have been particularly studied. This line of work focuses on adopting model-based methods [39, 40], which can capture human mobility patterns. However, most existing works along this line can only model simple transition patterns and cannot capture the multi-level and shifting nature of human mobility periodicity.

Overall, both the general time series data recovery methods, the mobility prediction methods and the existing mobility data recovery methods are incapable of tackling the instinctive challenges of the human mobility recovery problem. By contrast, we propose an attentional neural model based on graph neural network, which can model complex transition patterns among locations and capture the multi-level nature and shifting nature of human mobility periodicity simultaneously.

2.2 Neural Network on Graphs

Nowadays, neural network has been applied on graph structured data such as social network and knowledge bases. Extending the word2vec [23], an unsupervised algorithm DeepWalk [27] is proposed to learn representations of graph nodes based on random walk. Following DeepWalk, unsupervised network embedding algorithms LINE [31] and node2vec [11] are two most representative methods. Besides, the classical neural network CNN and RNN are also employed on graph structured data. For example, [8] designs a convolutional neural network that operates directly on graphs of arbitrary sizes and shapes. A scalable approach [13] chooses the convolutional architecture via a localized approximation of spectral graph convolutions, which is an efficient variant and can operate on graphs directly as well. However, these methods can only be implemented on undirected graphs. Previously, in form of recurrent neural networks, Recurrent Graph Neural Networks (RecGNNs) [28] are proposed to operate on directed graphs. As a modification of RecGNNs, gated GNN [17] is proposed to operate on directed graphs, which uses gated recurrent units and employs back-propagation through time (BPTT) to compute gradients. Since then, gated GNN has established new state-of-the-art benchmarks in a wide range of applications including script event prediction [18], situation recognition [16], image classification [22] and recommender system [29, 38, 41].

Previous studies [38, 41] have shown the ability of gated GNN to capture complex transition among nodes. In this paper, we leverage gated GNN to capture complex transition patterns among locations. To the best of our knowledge, we are the first to adopt the graph neural network to capture the complex location transition patterns in human mobility.

3 PRELIMINARIES

In this section, we first introduce the definitions we use in this paper followed by the formal definition of the investigated problem.

Definition 1 (Trajectory). *A trajectory is defined as a user's chronologically ordered location sequence in one day. Let* $\mathcal{T}_u^n : l_u^{n,1} \rightarrow l_u^{n,2} ... \rightarrow l_u^{n,t} ... \rightarrow l_u^{n,T}$ *denote user u's n-th day's trajectory, where* $l_u^{n,t} \in \mathcal{L}$ *is the most frequently visited location of t-th time slot for a given time interval (e.g., every 30 minutes) and T is the number of time slots. Note that if the location of time slot t is unobserved,* $l_u^{n,t}$ *is denoted by null, named missing location.*

Definition 2 (Current and Historical Trajectory). *Given a target day N and user u's trajectory* $\mathcal{T}_u^N$*, we define* $\mathcal{T}_u^N$ *as the user's current trajectory, and the historical trajectories are defined as u's trajectories in the past days, i.e.,*$\{\mathcal{T}_u^1, \mathcal{T}_u^2, ..., \mathcal{T}_u^{N-1}\}$.

Table 1: Summary of Notations.

Notation	Definition
u, U	A user and the set of users
$l, \mathcal{L}$	A location and the set of locations
$\mathcal{T}_u^n, \mathcal{T}_u^N$	The n-th historical trajectory of user u and the current trajectory of user u
$T, \mathcal{T}^{\mathcal{M}}$	The number of time slots in one trajectory and the set of missing time slots
$\mathbf{M}^{\mathbf{I}}, \mathbf{M}^{\mathbf{O}}$	The incoming matrix and outgoing matrix of the trajectory graph
$\mathbf{e}_l$	The embedding of location l
$\mathbf{e}_u^{n,t}$	The transition-aware location embedding of the t-th time slot in user u's n-th trajectory
$\check{\mathbf{e}}_t$	The temporal embedding of the t-th time slot
$\check{\mathbf{e}}_u^{n,t}$	The temporal-aware embedding of the t-th time slot in user u's n-th trajectory
$\alpha_{t,k,n}^{(h)}$	The similarity between the t-th time slot of the current trajectory and the k-th time slot of the n-th historical trajectory under the h-th head
$\hat{\mathbf{e}}_u^{n,t}$	The shift-aware embedding of the t-th time slot in user u's n-th trajectory
$\alpha_{n,t}$	The similarity between the t-th time slot of the current trajectory and the t-th time slot of the n-th historical trajectory
$\bar{\mathbf{e}}_u^t$	The final representation of user u's t-th time slot
$N_K(y_i), \tilde{N}_K(y_i)$	The nearest K locations for the target location y_i and other locations
$P_u^t(l), \mathbf{P}_u^t$	The normalized probability that user u visits location l at time slot t and the probability of all locations visited at time slot t
L_c, L_d	The cross-entropy loss and distance loss
ω, λ	The weight of distance loss and regularization
θ	General notation for model parameters

Now we give the formal definition of the investigated human mobility recovery problem.

Problem Statement. Given user u's current trajectory $\mathcal{T}_u^N$ with the historical trajectories $\{\mathcal{T}_u^1, \mathcal{T}_u^2, ..., \mathcal{T}_u^{N-1}\}$, we aim to recover the missing locations, i.e., $\forall$ null in $\mathcal{T}_u^N$ to rebuild the current day's complete trajectory.

4 THE PERIODICMOVE MODEL

To solve the above defined problem, we devise a novel neural attention model PeriodicMove, whose architecture is illustrated in Figure 2. In PeriodicMove, we first construct a directed graph for each trajectory and gated graph neural network is applied on the graph for complex location transition patterns modeling, followed by a temporal embedding layer for spatial-temporal dependency modeling. Then, in order to capture the shifting nature of human mobility periodicity, historical trajectories are fed into cross attention layer for shifting periodicity distilling based on current mobility status. In the following soft attention layer, multi-level nature of human mobility periodicity is captured by selecting the historical trajectories that are highly related to the current trajectory. Finally, to fuse

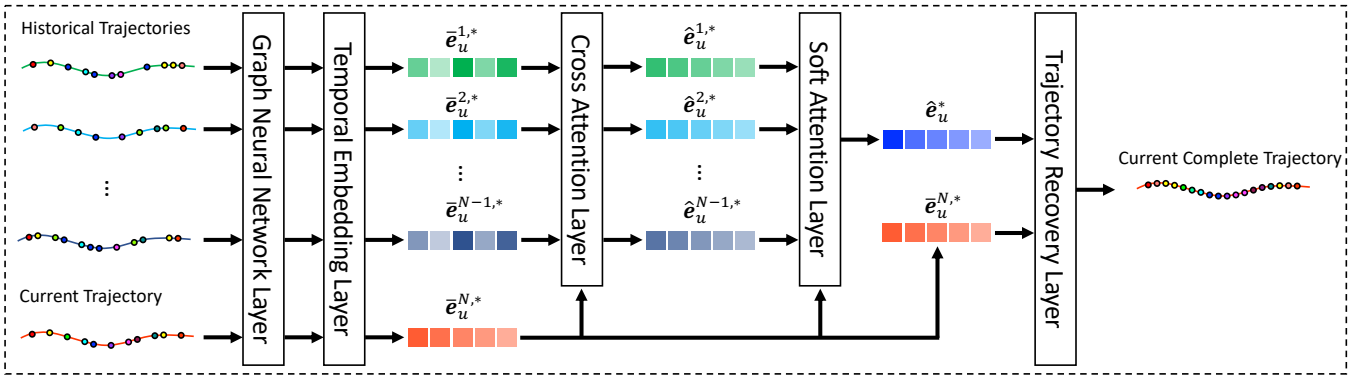


Figure 2: Main architecture of the PeriodicMove, where $*$ denotes all the time slots. In PeriodicMove, the historical trajectories and the current trajectories are first processed separately and then are fused to generate locations for missing time slots.

the features of historical trajectories and current trajectory, and generate missing locations as recovery, trajectory recovery layer is proposed as the final component.

4.1 Graph Neural Network Layer

Previous studies [38, 41] show that gated GNN is capable of capturing complex transition patterns among nodes, which makes gated GNN suit our problem. In the graph neural network layer, we process each trajectory separately to capture complex transition patterns hidden in each trajectory. Specifically, we first construct a directed graph for each trajectory and then gated GNN is applied on each directed graph to update the location embedding, which incorporates transition patterns into the model.

Graph Construction. The first part of graph neural network layer is to construct a meaningful graph for each historical trajectory and current trajectory. Given a trajectory $\mathcal{T}: l_1 \rightarrow l_2 ... \rightarrow l_T$, we treat each location l_i as a node and (l_{i-1}, l_i) as an edge which represents that a user visits location l_i after l_{i-1} in the trajectory $\mathcal{T}$. Therefore, each trajectory can be modeled as a directed graph. The graph structure is updated by promoting communication among different nodes. Specifically, let $\mathbf{M^I}, \mathbf{M^O} \in \mathbb{R}^{d \times d}$ denote weighted connections of incoming and outgoing edges in the trajectory graph, respectively. For example, considering a trajectory $\mathcal{T}: l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow l_2$, the corresponding graph and the matrix (i.e., $\mathbf{M^I}, \mathbf{M^O}$) are shown in Figure 3. Since several locations may appear in the trajectory repeatedly, we assign each edge with a normalized weight, which is calculated as the occurrence of the edge divided by the outdegree of that edge's start node.

Location Embedding Updating. Next, we present how to update embeddings of locations via gated graph neural network. We first embed each location $l \in \mathcal{L}$ into an unified low-dimensional latent space and the location vector $\mathbf{e}_l \in \mathbb{R}^d$ denotes a d-dimensional real-valued latent vector of location l.

For each location l at step s in the trajectory graph, given by the connection matrices $\mathbf{M}^I$ and $\mathbf{M}^O$, the information propagation can be formalized as:

$$\begin{aligned} \mathbf{a}_s = Concat(&\mathbf{M}_i^I([\mathbf{e}_1, ..., \mathbf{e}_N]\mathbf{W}_a^I + \mathbf{b}^I), \\ &\mathbf{M}_i^O([\mathbf{e}_1, ..., \mathbf{e}_N]\mathbf{W}_a^O + \mathbf{b}^O)), \end{aligned} \tag{1}$$

where $\mathbf{W}_a^I, \mathbf{W}_a^O \in \mathbb{R}^{d \times d}$ are learnable parameters. $\mathbf{b}^I, \mathbf{b}^O \in \mathbb{R}^d$ are the bias vectors. N is the number of unique locations in the trajectory. $\mathbf{M}_i^I, \mathbf{M}_i^O$ are the i-th row of incoming matrix and outgoing matrix corresponding to location l. $\mathbf{a}_s$ extracts the contextual information of neighborhoods for location l.

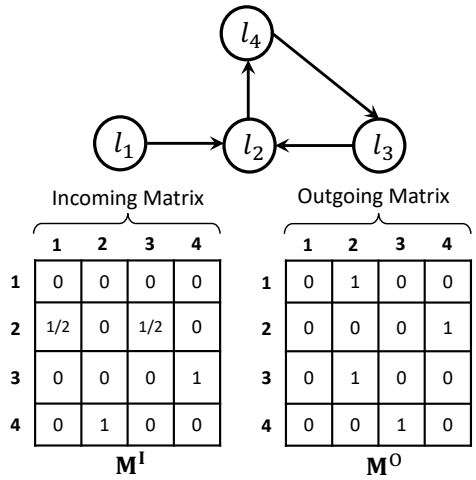


Figure 3: An example of a trajectory graph structure and the connection matrices $\mathbf{M^I}$ and $\mathbf{M^O}$.

Then, two gates, i.e., update gate $\mathbf{z}_s$ and reset gate $\mathbf{r}_s$, decide what information to be preserved and discarded respectively. After that, we construct the candidate state $\tilde{\mathbf{e}}_s$ by the previous state $\mathbf{e}_{s-1}$, the current state $\mathbf{a}_s$, and the reset gate $\mathbf{r}_s$ as described in Equation 4. The final state $\mathbf{e}_s$ is then the combination of the previous hidden state $\mathbf{e}_{s-1}$ and the candidate state $\tilde{\mathbf{e}}_s$, under the control of the update gate $\mathbf{z}_s$. After updating all nodes in trajectory graphs until convergence, we can obtain the transition-aware location embeddings.

$$\mathbf{z}_s = \sigma(\mathbf{W}_z\mathbf{a}_s + \mathbf{U}_z\mathbf{e}_{s-1}), \tag{2}$$

$$\mathbf{r}_s = \sigma(\mathbf{W}_r\mathbf{a}_s + \mathbf{U}_r\mathbf{e}_{s-1}), \tag{3}$$

$$\tilde{\mathbf{e}}_s = tanh(\mathbf{W}_h\mathbf{a}_s + \mathbf{U}_o(\mathbf{r}_s \odot \mathbf{e}_{s-1})), \tag{4}$$

$$\mathbf{e}_s = (1 - \mathbf{z}_s) \odot \mathbf{e}_{s-1} + \mathbf{z}_s \odot \tilde{\mathbf{e}}_s, \tag{5}$$

where $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h \in \mathbb{R}^{d \times 2d}, \mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_o \in \mathbb{R}^{d \times d}$ are learnable parameters. $\sigma(\cdot)$ represents the sigmoid function and $\odot$ denotes element-wise multiplication.

4.2 Temporal Embedding Layer

To integrate spatial-temporal dependency, we jointly embed the time and location into dense representation as the input of the following layers. Specifically, we set up embeddings for time slots. Following [32], for each time slot t, we generate its embedding as follows:

$$\begin{cases} \check{e}_t(2i) = sin(t/10000^{2i/d}), \\ \check{e}_t(2i+1) = cos(t/10000^{2i/d}), \end{cases} \tag{6}$$

where i denotes the i-th dimension. The time embedding vectors have the same dimension d with location embedding.

Finally, for the t-th time slot in user u's n-th trajectory, we obtain its temporal-aware representation, denoted as $\bar{\mathbf{e}}_u^{n,t} \in \mathbb{R}^d$, by calculating the sum of time and location embedding vectors:

$$\bar{\mathbf{e}}_u^{n,t} = \mathbf{e}_u^{n,t} + \check{\mathbf{e}}_t, \tag{7}$$

where $\mathbf{e}_u^{n,t}$ is the transition-aware location embedding of the t-th time slot in user u's n-th trajectory.

4.3 Cross Attention Layer

Periodicity shifting is a common feature existed in human mobility data. For example, during weekday, people may arrive at workplace at different time due to some environmental factors (e.g., traffic jam). To detect the shifting behavior existed in each historical trajectory, one good solution is to process each historical trajectory separately, compare the current mobility status in current trajectory with all historical mobility statuses in each historical trajectory, aggregate all historical information according to their similarity and obtain a shift-aware embedding for each time slot of each historical trajectory.

Multi-head attention is proposed by [33] to stabilize the learning process and incorporate more information into the network. Here we also implement H independent heads to calculate the embeddings. Specifically, we define the similarity between the t-th time slot of the current trajectory and the k-th time slot of the n-th historical trajectory, i.e., between $\bar{\mathbf{e}}_u^{N,t}$ and $\bar{\mathbf{e}}_u^{n,k}$, under head h as follows:

$$\alpha_{t,k,n}^{(h)} = \frac{exp(\phi^{(h)}(\bar{\mathbf{e}}_u^{N,t}, \bar{\mathbf{e}}_u^{n,k}))}{\sum_{g=1}^{T} exp(\phi^{(h)}(\bar{\mathbf{e}}_u^{N,t}, \bar{\mathbf{e}}_u^{n,g}))}, \tag{8}$$

$$\phi^{(h)}(\bar{\mathbf{e}}_u^{N,t}, \bar{\mathbf{e}}_u^{n,k}) = < \mathbf{W}_Q^{(h)} \bar{\mathbf{e}}_u^{N,t}, \mathbf{W}_K^{(h)} \bar{\mathbf{e}}_u^{n,k} >, \tag{9}$$

where $\mathbf{W}_Q^{(h)}, \mathbf{W}_K^{(h)} \in \mathbb{R}^{d \times d}$ are the transformation matrices under head h and $<,>$ denotes the inner product function.

Then, for the t-th time slot of user u's n-th historical trajectory, we obtain its shift-aware embedding $\check{\mathbf{e}}_u^{n,t}$ by aggregating the information from all the time slots of the n-th historical trajectory by the similarity $\boldsymbol{\alpha}_{t,*,n}^{(h)}$.

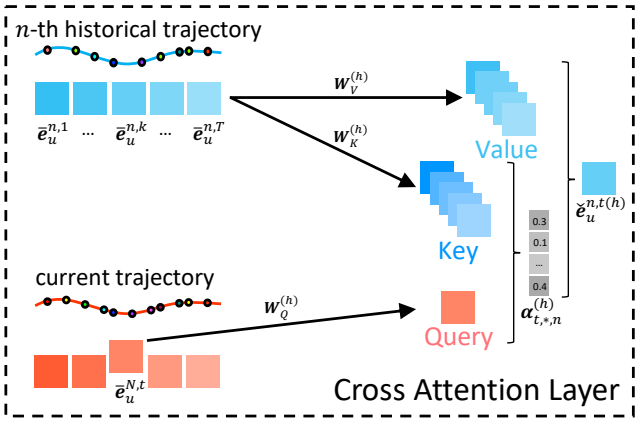


Figure 4: The architecture of proposed cross attention layer under one head, where the new representation is a combination of embeddings of value time slots conditioned on attention weights $\boldsymbol{\alpha}_{t,*,n}$ from query and key time slots. Note that superscript n denotes the n-th historical trajectory while N denotes the current trajectory.

$$\begin{aligned} \check{\mathbf{e}}_u^{n,t(h)} &= \sum_{g=1}^{T} \alpha_{t,g,n}^{(h)} (\mathbf{W}_V^{(h)} \bar{\mathbf{e}}_u^{n,g}), \\ \check{\mathbf{e}}_u^{n,t} &= \check{\mathbf{e}}_u^{n,t(1)} \| \check{\mathbf{e}}_u^{n,t(2)} \| \dots \| \check{\mathbf{e}}_u^{n,t(H)}, \end{aligned} \tag{10}$$

where $\mathbf{W}_V^{(h)} \in \mathbb{R}^{d \times d}$ is also a transformation matrix under head h.

Finally, a standard residual connection is added to preserve current mobility information and obtain a shift-aware embedding for each time slot of historical trajectories.

$$\hat{\mathbf{e}}_u^{n,t} = ReLU(\mathbf{W}_1 \check{\mathbf{e}}_u^{n,t} + \bar{\mathbf{e}}_u^{N,t}) \tag{11}$$

4.4 Soft Attention Layer

Human mobility's periodicity of daily routines, weekly leisure and yearly festivals can be completely different, which demonstrates multi-level nature of human mobility periodicity. To capture multi-level periodicity of human mobility, we need an auto-selector to choose the highly related historical records of current mobility status from the historical trajectories as the periodicity representation.

Specifically, we leverage soft attention mechanism to calculate the similarity between the corresponding time slot of the current trajectory and each historical trajectory, and aggregate time slot embeddings of historical trajectories based on the similarity to obtain the periodicity representation for each time slot.

$$\begin{aligned} \alpha_{n,t} &= \mathbf{q}^T \sigma(\mathbf{W}_2 \hat{\mathbf{e}}_u^{n,t} + \mathbf{W}_3 \bar{\mathbf{e}}_u^{N,t} + \mathbf{c}), \\ \hat{\mathbf{e}}_u^t &= \sum_{n=1}^{N-1} \alpha_{n,t} \hat{\mathbf{e}}_u^{n,t}, \end{aligned} \tag{12}$$

where $\hat{\mathbf{e}}_u^{n,t}$ is the shift-aware embedding of the t-th time slot in the n-th historical trajectory, $\bar{\mathbf{e}}_u^{N,t}$ is the current mobility embedding of the t-th time slot in the current trajectory, parameter $\mathbf{q} \in \mathbb{R}^d$ and

$\mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{d \times d}$ control the weight of historical trajectories and current trajectory respectively.

4.5 Trajectory Recovery Layer

To generate the final representation of the t-th time slot denoted by $\bar{\mathbf{e}}_u^t$, we apply a linear transformation over the concatenation of the periodicity representation $\hat{\mathbf{e}}_u^t$ and current mobility representation $\bar{\mathbf{e}}_u^{N,t}$.

$$\bar{\mathbf{e}}_u^t = \mathbf{W}_4[\hat{\mathbf{e}}_u^t \| \bar{\mathbf{e}}_u^{N,t}] \tag{13}$$

Once we obtained $\bar{\mathbf{e}}_u^t$, we are able to calculate the probability that user u visits location l at time slot t as follows:

$$P_u^t(l) = \frac{< \bar{\mathbf{e}}_u^t, \mathbf{e}_l >}{\sum_{k \in \mathcal{L}} < \bar{\mathbf{e}}_u^t, \mathbf{e}_k >}, \tag{14}$$

where $P_u^t(l)$ denotes the normalized probability that user u visits location l in current trajectory's t-th time slot. In practice, the location with the maximum probability is identified as the missing location.

4.6 Training

Existing trajectory recovery models usually adopt the following cross-entropy loss function for model training:

$$L_c = -\sum_{u \in U} \sum_{t \in \mathcal{T}^{\mathcal{M}}} \mathbf{y}_u^t log(\mathbf{P}_u^t), \tag{15}$$

where $\mathbf{y}_u^t$ is the one-hot representation of user u's location in current trajectory's t-th time slot, $\mathbf{P}_u^t \in \mathbb{R}^{|\mathcal{L}|}$ denotes the probability of all locations visited in current trajectory's t-th time slot and $\mathcal{T}^{\mathcal{M}}$ denotes the set of missing time slots.

However, in highly sparse scenarios, the proposed cross-entropy loss cannot capture the spatial proximity well which is an important feature for human mobility recovery. So we propose a distance loss to incorporate spatial proximity information into the model. And Noise Contrastive Estimation (NCE) is adopted to accelerate the training. Specifically, for a target location y_i, we find the nearest K locations (i.e., $N_K(y_i)$) and treat these locations as positive neighbors and others as negative ones (i.e., $\tilde{N}_K(y_i)$). Then, we randomly choose one positive sample $p \in N_K(y_i)$ and one negative sample $q \in \tilde{N}_K(y_i)$, and calculate the NCE-based distance loss as follows:

$$L_d = \sum_{i=1}^{|y|} w_{py_i} \, max(||\mathbf{e}_{y_i} - \mathbf{e}_p||_2 - ||\mathbf{e}_{y_i} - \mathbf{e}_q||_2 + m, 0), \tag{16}$$

$$w_{py_i} = \frac{exp(-Dist(p, y_i))}{\sum_{x \in N_K(y_i)} exp(-Dist(x, y_i))}, \tag{17}$$

where y is the set of all unique locations in one training batch, m is a margin hyper-parameter, $|| \cdot ||_2$ is the 2-norm of vector and $Dist(\cdot)$ is the distance between two locations.

Finally, we have the spatial-aware loss function as follows:

$$L_{PeriodicMove} = L_c + \omega L_d + \lambda \|\theta\|^2, \tag{18}$$

where ω is distance weight that balances cross entropy loss and distance loss, and λ is a hyper-parameter that controls the power of regularization.

Training algorithm is illustrated in Algorithm 1, and the process is done through batch gradient descent over shuffled mini-batches across AdamW [20]. In addition, our model is implemented by Python and Pytorch [26]. All the models are implemented on a linux server with 48-cores Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz 256GB RAM and one TITAN Xp GPU.

Algorithm 1: Training Algorithm for PeriodicMove

Input: Trajectory sets $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_U\}$
Output: Trained Model θ
// Construct training instances $\mathcal{D}$
// Train the model: initialize the model parameters θ
for $i \in$ *{1, 2,..., EPOCH}* **do**
  Shuffle the training instances $\mathcal{D}$ into mini-batches
  for $j \in$ *{1, 2,..., BATCH NUM}* **do**
    Construct trajectory graphs according to section 4.1
    Forward propagation and calculate the cross-entropy loss L_c according to Equation 15
    Select unique locations and calculate the distance loss L_d according to Equation 16
    Calculate the loss function according to Equation 18 and Update θ by minimizing $L_{PeriodicMove}$
  Stop training when criteria is met
Output trained model θ

5 EXPERIMENT

5.1 Datasets

- **Geolife**[1]: This dataset [48–50] is collected in (Microsoft Research Asia) Geolife project by 182 users from April 2007 to August 2012. A GPS trajectory of this dataset is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude.
- **Foursquare**[2]: This dataset [42] is collected from Foursquare API from April 2012 to February 2013. Every record in the dataset consists of user ID, timestamp, GPS location and POI ID. We normalize the timestamp into one week while keeping the original order of the trajectory.

Table 2: Basic statistics of mobility datasets.

Dataset	City	Duration	#Users	#Loc	#Traj Pair
Geolife	Beijing	5 years	83	1124	3912
Foursquare	Tokyo	11 months	841	1411	2286

Pre-processing: To represent the location, we adopt a simple strategy that is used commonly in spatial data analytics [4, 7, 12, 35, 45, 46], i.e., we partition the geographical space into grid cells of equal size and all the locations within the same grid cell are considered to be the same location. Specifically, each grid cell has the side length of 500m for both datasets. Following [3], we set time interval as 30 minutes. For fair comparison, we first filter out the locations appearing less than 5 times and then filter out trajectories

[1] https://www.microsoft.com/en-us/research/project/geolife-building-social-networks-using-human-location-history/
[2] https://sites.google.com/site/yangdingqi/home/foursquare-dataset/

Table 3: Overall performance comparison in terms of Recall@K, Distance@K and MAP.

		Recall@1	Recall@5	Recall@10	Distance@1	Distance@5	Distance@10	MAP
Geolife	Top	0.1148	0.2451	0.3166	7863	6259	5176	0.1812
	Markov	0.1417	0.3263	0.3974	6909	4974	4259	0.2304
	PMF	0.1941	0.3436	0.4059	6506	4389	3555	0.2752
	LSTM	0.2086	0.3917	0.4720	6318	3928	3068	0.2965
	BiLSTM	0.2285	0.4538	0.5773	6209	3620	2255	0.3298
	DeepMove	0.3045	0.5380	0.6371	5370	2052	1358	0.4131
	AttnMove	0.3920	0.6696	0.7213	5342	2007	975	0.5046
	PeriodicMove	**0.4199**	**0.6893**	**0.7681**	**4209**	**1443**	**863**	**0.5385**
	%Improv.	**7.1%**	**2.9%**	**6.5%**	**21.2%**	**28.1%**	**11.5%**	**6.7%**
Foursquare	Top	0.0865	0.1673	0.2268	8427	4919	3483	0.1347
	Markov	0.1090	0.2010	0.2575	8345	4402	3125	0.1792
	PMF	0.1215	0.2468	0.2887	8116	3971	3229	0.2358
	LSTM	0.1393	0.2540	0.3143	7913	3804	2801	0.2519
	BiLSTM	0.2323	0.3968	0.4703	6206	2745	1849	0.3154
	DeepMove	0.2612	0.4631	0.5337	5189	2648	1649	0.3789
	AttnMove	0.2975	0.5172	0.5746	4942	2396	1482	0.4078
	PeriodicMove	**0.3125**	**0.5534**	**0.6264**	**4704**	**1758**	**1197**	**0.4245**
	%Improv.	**5.0 %**	**7.0%**	**9.0%**	**4.8%**	**26.6%**	**19.2 %**	**4.1%**

with less than 36 time slots and the users with only one trajectory. The final detailed statics of two mobility datasets are summarized in Table 2.

5.2 Baselines

We compare the proposed method with several representative baselines. Among them, the first three are directly based on our knowledge about human mobility and the last four are the state-of-the-art deep learning models which can extract more complex mobility features:

- **Top**: This is a simple counting-based method, which directly uses the most popular location in the training set as recovery for each user.
- **Markov** [10]: This is a widely used method for human mobility prediction, which regards all the visited locations as states and builds a transition matrix to capture the first order transition probabilities between them.
- **PMF** [24]: Based on user location matrix, this is one of the state-of-the-art models for conventional collaborative filtering.
- **LSTM** [19]: This is a deep learning model which models the forward sequential transitions of mobility by recurrent neural network and uses the prediction for next time slot as recovery.
- **BiLSTM** [44]: It extends LSTM by bidirectional RNN to consider the spatial-temporal constraints given by all observed locations.
- **DeepMove** [9]: This is a state-of-the-art model which jointly models user preferences and spatial-temporal dependency for next location prediction. We use the prediction result for recovery.
- **AttnMove** [40]: This is the latest trajectory recovery method which leverages various attention mechanisms to model the regularity and periodical patterns of user's mobility.

5.3 Experimental Settings

To evaluate the performance, we mask some time slots as ground-truth to recover. Since about 20% locations are missing, we randomly mask 10 time slots per day for Geolife dataset and Foursquare dataset. We sort each user's trajectory by time and take the first 60% as the training set, the following 20% as the validation set and the remaining 20% as the test set.

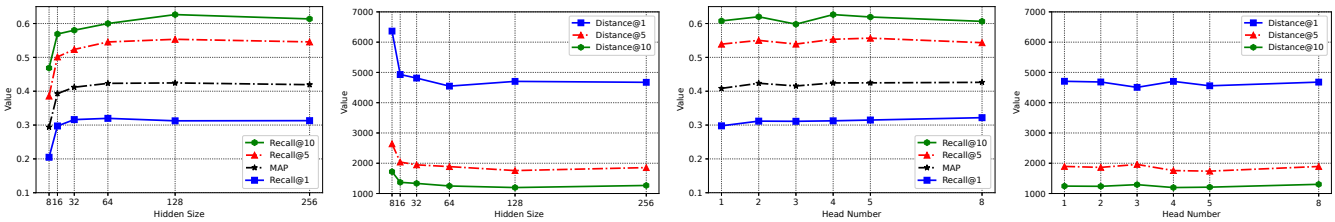
To prevent the model from overfitting, we apply dropout mechanism in trajectory recovery layer. The dropout rate is set to be 0.3. Besides, distance weight ω is initialized as 0.1, regularization factor λ is initialized as 1×10^{-5} and margin hyper-parameter m is initialized as 1.

We employ the widely used metrics **Recall@K** and Mean Average Precision (**MAP**). Recall@K is 1 if the ground truth location appears in the top-K ranked list; otherwise is 0. The final Recall@K is the average value over all testing instances. MAP is a global evaluation for ranking tasks, so we use it to evaluate the quality of the whole ranked lists including all locations. The larger the values of these two metrics are, the better the performance will be. We also make use of the metric of **Distance@K**, which is the smallest geographical distance between the centers of locations in the top-K ranked list and the ground truth location. The final Distance@K is the average value over all testing instances. The smaller the value of Distance@K is, the better the performance will be. We report Recall@K and Distance@K with K = 1, 5 and 10 in our experiments.

5.4 Experiment Results

5.4.1 Overall Performance. From the results shown in Table 3, several observations can be made:

Firstly, Rule-based methods fail to achieve high accuracy with the worst performance for all evaluation metrics on both datasets. Though utilizing knowledge about human mobility are helpful for recovery, their performances are not acceptable because they fail



(a) Impart of hidden size d (H = 4)

(b) Impart of head number H (d = 128)

Figure 5: Sensitivity of hyper-parameters.

to capture the complex transition patterns among locations and periodical patterns of human mobility.

Secondly, RNN-based methods perform better than rule-based methods because they can model simple transition patterns among locations. And bidirectional ones perform better than unidirectional ones, which indicates the importance of spatial-temporal constraints for human mobility recovery. State-of-the-art deep learning methods including DeepMove and AttnMove achieve satisfactory performance because they can capture simple location transition patterns and single level periodicity of human mobility. However, our PeriodicMove achieves further performance gain over the best deep learning methods because PeriodicMove can capture complex location transition patterns and multi-level periodicity of human mobility simultaneously.

Thirdly, PeriodicMove outperforms all the baselines for all evaluation metrics on both datasets. Specifically, Recall of PeriodicMove outperforms the best baseline AttnMove by 2.9%∼7.1% on Geolife dataset and 5.0%∼9.0% on Foursquare dataset respectively. Distance of PeriodicMove outperforms the best baseline AttnMove by 11.5%∼28.1% on Geolife dataset and 4.8%∼26.6% on Foursquare dataset respectively. MAP of PeriodicMove outperforms the best baseline AttnMove by 6.7% on Geolife dataset and 4.1% on Foursquare dataset respectively. These great improvements indicate that our proposed PeriodicMove can well model location transition patterns and periodicity of human mobility.

To conclude, our PeriodicMove model achieves preferable results compared with both rule-based and deep learning-based methods. This verifies our model's effectiveness in capturing complex transition patterns among locations and periodicity of human mobility, which demonstrates its powerful ability to rebuild fine-grained individual trajectories.

5.4.2 Ablation Study. We analyze the effects of each model component. We create ablation by removing them one by one. Specifically, we remove graph neural network layer, temporal embedding layer, cross attention layer, replace weighted average with arithmetic mean in soft attention layer and eliminate distance loss, respectively.

As expected, PeriodicMove outperforms all the ablations, which indicates the importance of each component in improving recovery accuracy. Specifically, the performance drops the most significantly when we remove soft attention layer, which captures multi-level periodicity of human mobility. This is because soft attention layer can identify historical trajectories that are highly related to the current trajectory. By giving higher weight to the highly related historical trajectories, the noise from unrelated historical trajectories can be reduced, and thus improve the model performance. It is worth noting that the performance drop on recall@10 is not significant when we remove temporal embedding layer. A plausible reason is that the sequential order information that the temporal embedding layer hopes to convey has been partially modeled by graph neural network layer during the construction of directed graphs, which again verifies the effectiveness of graph neural network layer.

Table 4: Impact of each component on Foursquare dataset, where Δ denoted the performance decline.

Ablation	Recall@10 (Δ)	Distance@10 (Δ)	MAP (Δ)
Graph Neural Network Layer	0.6068 (-3.1%)	1286 (-7.4%)	0.4193 (-1.2%)
Temporal Embedding Layer	0.6114 (-2.4%)	1308 (-9.3%)	0.4184 (-1.4%)
Cross attention Layer	0.6050 (-3.4%)	1285 (-7.4%)	0.3983 (-6.2%)
Soft attention Layer	**0.5598 (-10.6%)**	**1427 (-19.2%)**	**0.3764 (-11.3%)**
Distance Loss	0.5984 (-4.5%)	1276 (-6.6%)	0.4237 (-0.2%)

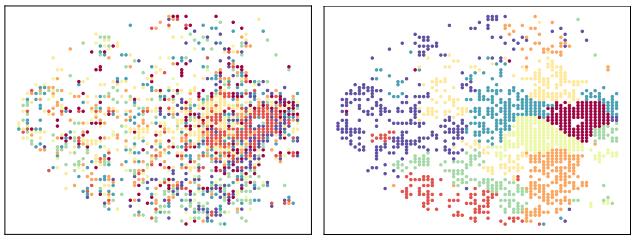
5.4.3 Sensitivity of hyper-parameters. We also investigate the sensitivity of two important hyper-parameters including hidden size d and head number H. Here we only report the results on Foursquare dataset, we observe a similar trend on Geolife dataset.

Firstly, we observe the performance change by tuning the hidden size d in the range of {8, 16, 32, 64, 128, 256}. From the result presented in Figure 5(a), we can see: as the hidden size increases, the performance is gradually improved, and when it is larger than a value, the performance starts to decline. In most cases, the value is 128, which is also the reason why we select default hidden size as 128. This observation actually agrees with many previous studies [5, 14, 43]. One reason can be that: in our dataset, a small number of parameters are enough for capturing transition patterns among locations and periodicity of human mobility, using redundant dimensions will increase the model complexity and force the model to overfit the training set, which may degrade our model's generalization capability on the test set.

Then, we tune the head number H in the range of {1, 2, 3, 4, 5, 8} and Figure 5(b) shows the performance change. As we can see, the change of performance with regard to head number does not demonstrate a clear trend in most cases, which implies the impart of head number is not significant. Considering more heads leads to stronger model expressiveness and more computational cost, to

make a compromise between performance and efficiency, we finally fix the head number at 4.

5.4.4 Visualization Analysis. Apart from the above quantitative analysis, we also conduct a visualization on Foursquare dataset to verify the effectiveness of distance loss in capturing spatial correlation among the locations.



(a) Result without distance loss (b) Result with distance loss

Figure 6: Visualization results.

We cluster the locations by using their embedding $\mathbf{e}_l$ as features via k-means with Euclidean distance and the same color indicates the same cluster. Figure 6(a) and Figure 6(b) demonstrate the clustering result of model trained without distance loss and with distance loss, respectively. As we can see, in the cluster result trained with distance loss, adjacent locations generally share the same color indicating they are also closed in the embedding space, which demonstrates that the spatial adjacent relation has been modeled. But in the cluster result trained without distance loss, no clear features can be observed, which verifies the important role of distance loss in capturing spatial correlation among the locations.

5.4.5 Robustness Analysis. We also conduct experiments to evaluate the robustness of PeriodicMove towards data sparsity problem. To simulate the effect of data sparsity, we increase the percentage of missing locations in historical trajectories from 20% to 80%. We compare our model PeriodicMove with state-of-the-art model AttnMove and the result is demonstrated in Table 5.

Table 5: Performance w.r.t missing ratios on Foursquare dataset.

Missing Rate		20%	40%	60%	80%
AttnMove	Recall@10	0.5596	0.5501	0.5357	0.5269
	Dist@10	1579	1668	1874	1937
	MAP	0.4014	0.3942	0.3809	0.3783
PeriodicMove	Recall@10	0.6132	0.6082	0.5961	0.5827
	Dist@10	1244	1283	1361	1380
	MAP	0.4194	0.4133	0.3987	0.3944

From the result, we can see: as the missing rate increases, both PeriodicMove and AttnMove's performances on all evaluation metrics drop. However, PeriodicMove still outperforms AttnMove on all missing rates consistently. In fact, the performance of our PeriodicMove on Foursquare dataset with 80% missing rate is better than the performance of AttnMove on Foursquare dataset with 40% missing rate, which demonstrates the robustness of our proposed model towards data sparsity problem. This is because AttnMove aggregates all the historical trajectories into one trajectory by simply picking the most frequently visited location for each time slot, which ignores rich information hidden in historical trajectories. In contrast, PeriodicMove processes each historical trajectory separately to capture the location transition patterns and periodicity of human mobility hidden in each historical trajectory, and aggregates these information in the final trajectory recovery layer. Moreover, the proposed spatial-aware loss function incorporates spatial proximity into the model optimization, which further alleviates the data sparsity problem.

6 CONCLUSION

In this paper, we propose PeriodicMove, a neural attention model based on graph neural network for human mobility recovery from lengthy and sparse individual trajectories. We leverage graph neural network for complex location transition patterns modeling. We design a cross attention layer to capture shifting behavior of human mobility periodicity. We use a soft attention layer to capture the multi-level nature of human mobility periodicity. To handle the data sparsity problem, we design a spatial-aware loss function to incorporate the spatial proximity into the model. Extensive experiment results on two real-life mobility datasets demonstrate the superiority of PeriodicMove compared with the state-of-the-art baselines.

As for future work, we plan to incorporate semantic information like point of interests [36, 37] into the model so that our model can not only recover the missing locations, but also understand the underlying motivation of user's movement.

ACKNOWLEDGMENTS

This work is partially supported by Natural Science Foundation of China (No. 61972069, 61836007 and 61832017).

REFERENCES

[1] Layth C Alwan and Harry V Roberts. 1988. Time-series modeling for statistical process control. *Journal of business & economic statistics* 6, 1 (1988), 87–95.
[2] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. Brits: Bidirectional recurrent imputation for time series. *arXiv preprint arXiv:1805.10572* (2018).
[3] Guangshuo Chen, Aline Carneiro Viana, Marco Fiore, and Carlos Sarraute. 2019. Complete trajectory reconstruction from sparse mobile phone data. *EPJ Data Science* 8, 1 (2019), 30.
[4] Meng Chen, Yan Zhao, Yang Liu, Xiaohui Yu, and Kai Zheng. 2020. Modeling spatial trajectories with attribute representation learning. *TKDE* (2020).
[5] Xu Chen, Yongfeng Zhang, and Zheng Qin. 2019. Dynamic explainable recommendation based on neural attentive models. In *AAAI*, Vol. 33. 53–60.
[6] Yue Cui, Hao Sun, Yan Zhao, Hongzhi Yin, and Kai Zheng. 2021. Sequential-knowledge-aware Next POI Recommendation: A Meta-learning Approach. *TOIS* (2021).
[7] Liwei Deng, Hao Sun, Rui Sun, Yan Zhao, and Han Su. 2021. Efficient and Effective Similar Subtrajectory Search: A Spatial-aware Comprehension Approach. *TIST* (2021).
[8] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292* (2015).
[9] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. Deepmove: Predicting human mobility with attentional recurrent networks. In *WWW*. 1459–1468.
[10] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. 2012. Next place prediction using mobility markov chains. In *Proceedings of the first workshop on measurement, privacy, and mobility*. 1–6.

[11] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *KDD*. 855–864.

[12] Ralf Hartmut Güting and Markus Schneider. 1995. Realm-based spatial data types: The ROSE algebra. *The VLDB Journal* 4, 2 (1995), 243–286.

[13] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[14] Dongsheng Li, Chao Chen, Qin Lv, Junchi Yan, Li Shang, and Stephen Chu. 2016. Low-rank matrix approximation with stability. In *ICML*. PMLR, 295–303.

[15] Li Li, Yuebiao Li, and Zhiheng Li. 2013. Efficient missing data imputing for traffic flow by considering temporal and spatial dependence. *Transportation research part C: emerging technologies* 34 (2013), 108–120.

[16] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. 2017. Situation recognition with graph neural networks. In *ICCV*. 4173–4182.

[17] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).

[18] Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction. *arXiv preprint arXiv:1805.05081* (2018).

[19] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*, Vol. 30.

[20] Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam. (2018).

[21] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, and Xiaojie Yuan. 2018. Multivariate time series imputation with generative adversarial networks. In *NIPS*. 1603–1614.

[22] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. 2016. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844* (2016).

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546* (2013).

[24] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *NIPS* 20 (2007), 1257–1264.

[25] Steffen Moritz and Thomas Bartz-Beielstein. 2017. imputeTS: time series missing value imputation in R. *R J.* 9, 1 (2017), 207.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[29] Hao Sun, Zijian Wu, Yue Cui, Liwei Deng, Yan Zhao, and Kai Zheng. 2021. Personalized Dynamic Knowledge-Aware Recommendation with Hybrid Explanations. In *DASFAA*. 148–164.

[30] Ke Sun, Tieyun Qian, Tong Chen, Yile Liang, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2020. Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In *AAAI*, Vol. 34. 214–221.

[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[34] Jingyuan Wang, Ning Wu, Xinxi Lu, Xin Zhao, and Kai Feng. 2019. Deep trajectory recovery with fine-grained calibration using Kalman filter. *TKDE* (2019).

[35] Zheng Wang, Cheng Long, Gao Cong, and Yiding Liu. 2020. Efficient and effective similar subtrajectory search with deep reinforcement learning. *arXiv preprint arXiv:2003.02542* (2020).

[36] Fei Wu and Zhenhui Li. 2016. Where did you go: Personalized annotation of mobility records. In *CIKM*. 589–598.

[37] Fei Wu, Zhenhui Li, Wang-Chien Lee, Hongjian Wang, and Zhuojie Huang. 2015. Semantic annotation of mobility data using social media. In *WWW*. 1253–1263.

[38] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*, Vol. 33. 346–353.

[39] Dongbo Xi, Fuzhen Zhuang, Yanchi Liu, Jingjing Gu, Hui Xiong, and Qing He. 2019. Modelling of bi-directional spatio-temporal dependence and users' dynamic preferences for missing poi check-in identification. In *AAAI*, Vol. 33. 5458–5465.

[40] Tong Xia, Yunhan Qi, Jie Feng, Fengli Xu, Funing Sun, Diansheng Guo, and Yong Li. 2021. AttnMove: History Enhanced Trajectory Recovery via Attentional Network. *arXiv preprint arXiv:2101.00646* (2021).

[41] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *IJCAI*, Vol. 19. 3940–3946.

[42] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. 2014. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 1 (2014), 129–142.

[43] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *CIKM*. 1449–1458.

[44] Jing Zhao, Jiajie Xu, Rui Zhou, Pengpeng Zhao, Chengfei Liu, and Feng Zhu. 2018. On prediction of user destination by sub-trajectory understanding: A deep learning based approach. In *CIKM*. 1413–1422.

[45] Yan Zhao, Shuo Shang, Yu Wang, Bolong Zheng, Quoc Viet Hung Nguyen, and Kai Zheng. 2018. Rest: A reference-based framework for spatio-temporal trajectory compression. In *SIGKDD*. 2797–2806.

[46] Kai Zheng, Yan Zhao, Defu Lian, Bolong Zheng, Guanfeng Liu, and Xiaofang Zhou. 2019. Reference-based framework for spatio-temporal trajectory compression and query processing. *TKDE* 32, 11 (2019), 2227–2240.

[47] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *TIST* 5, 3 (2014), 1–55.

[48] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *UbiComp*. 312–321.

[49] Yu Zheng, Xing Xie, and Wei-Ying Ma. 2010. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.

[50] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*. 791–800.

[51] Hao Zhou, Yan Zhao, Junhua Fang, Xuanhao Chen, and Kai Zeng. 2019. Hybrid route recommendation with taxi and shared bicycles. *Distributed and Parallel Databases* (2019), 1–21.