# Robust Location Prediction over Sparse Spatiotemporal Trajectory Data: Flashback to the Right Moment!

BANGCHAO DENG and DINGQI YANG\*, University of Macau, China BINGQING QU, BNU-HKBU United International College, China BENJAMIN FANKHAUSER, Bern University of Applied Sciences, Switzerland PHILIPPE CUDRE-MAUROUX, University of Fribourg, Switzerland

As a fundamental problem in human mobility modeling, location prediction forecasts a user's next location based on historical user mobility trajectories. Recurrent Neural Networks (RNNs) have been widely used to capture sequential patterns of user visited locations for solving location prediction problems. Due to the sparse nature of real-world user mobility trajectories, existing techniques strive to improve RNNs by incorporating spatiotemporal contexts into the recurrent hidden state passing process of RNNs using context-parameterized transition matrices or gates. However, such a scheme mismatches universal spatiotemporal mobility laws and thus cannot fully benefit from rich spatiotemporal contexts encoded in user mobility trajectories. Against this background, we propose Flashback++, a general RNN architecture designed for modeling sparse user mobility trajectories. It not only leverages rich spatiotemporal contexts to search past hidden states with high predictive power, but also learns to optimally combine them via a hidden state re-weighting mechanism, which significantly improves the robustness of the models against different settings and datasets. Our extensive evaluation compares Flashback++ against a sizable collection of state-of-the-art techniques on two real-world LBSN datasets and one on-campus mobility dataset. Results show that Flashback++ not only consistently and significantly outperforms all baseline techniques by 20.56% to 44.36%, but also achieves better robustness of location prediction performance against different model settings (different RNN architectures and numbers of hidden states to flash back), different levels of trajectory sparsity, and different train-testing splitting ratios than baselines, yielding an improvement of 31.05% to 94.60%.

 ${\tt CCS\ Concepts: \bullet\ Computing\ methodologies} \rightarrow {\tt Neural\ networks; \bullet\ Information\ systems} \rightarrow {\tt Location\ based\ services}.$ 

Additional Key Words and Phrases: Location prediction, Sparse trajectory, User mobility, Recurrent neural networks

#### **ACM Reference Format:**

## 1 INTRODUCTION

Human mobility modeling is one of the key problems in understanding human dynamics [16, 49]. A high-quality mobility model can serve as a fundamental ingredient for developing many smart city

Authors' addresses: Bangchao Deng; Dingqi Yang\*, University of Macau, Avenue of University, Macao SAR, China, {mc14 969,dingqiyang}@um.edu.mo; Bingqing Qu, BNU-HKBU United International College, China, bingqingqu@uic.edu.cn; Benjamin Fankhauser, Bern University of Applied Sciences, Switzerland, benjamin.fankhauser@bfh.ch; Philippe Cudre-Mauroux, University of Fribourg, Switzerland, philippe.cudre-mauroux@unifr.ch.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Association for Computing Machinery.

0004-5411/2018/8-ART111 \$15.00

https://doi.org/XXXXXXXXXXXXXXX

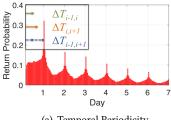
111:2 Deng et al.

applications, such as personalized location recommendation [18, 32, 50], urban planning [47, 48] and taxi ride-sharing scheduling [33, 35]. In this context, one key task of user mobility modeling is to predict an individual's next location based on users' historical mobility trajectories [1]. Traditional methods often resort to various user mobility features — either hand-crafted features such as historical visit counts [39], or automatically-learnt features using graph embedding techniques [55] — to capture user mobility patterns. However, by generating static features from historical data, these techniques predict user locations without really considering the sequential patterns of user mobility, which have indeed been shown as an important clue for location prediction [30].

In the current literature, Recurrent Neural Networks (RNNs) have been shown as a successful tool to model sequential data, and thus started to be used also for user mobility modeling [68]. However, classical RNN architectures, such as vanilla RNN, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were originally designed for language modeling to learn from word sequences (sentences), which differs from user mobility trajectories that are often *sparse and incomplete*. Specifically, on Location Based Social Networks (LBSNs), which have been widely used as a primary data source for user mobility modeling [58], a user's mobility trajectory is stored as a sequence of check-ins, where each check-in represents the user's presence at a specific Point of Interests (POIs) such as a restaurant or a gym, at a specific time. As users share their check-ins within their social circles on a *voluntary* basis on LBSNs, such mobility trajectories are often sparse. For example, on our collected Foursquare dataset, we find that the median time between successive check-ins is about 16.72 hours. Such sparsity and incompleteness of input sequences hinder the application of RNNs to the location prediction problem [11, 56]. To handle such sparse mobility trajectories, existing works strive to incorporate spatiotemporal factors into RNN architectures, as spatiotemporal contexts have been shown as strong predictors for user mobility prediction [56, 62].

To achieve this goal, the most popular scheme used by existing work is to incorporate spatiotemporal factors into the recurrent hidden state passing process of RNNs. Specifically, given a user mobility trajectory represented as a sequence of POIs  $\{..., p_{i-1}, p_i, p_{i+1}, ...\}$ , this scheme first computes the temporal and spatial distances between the previous check-in and the current check-in, denoted as  $\Delta T_{i-1,i}$ ,  $\Delta D_{i-1,i}$ , respectively, and then feeds them as additional inputs to the RNN unit, i.e.,  $h_i = \mathcal{F}(p_i, h_{i-1}, \Delta T_{i-1,i}, \Delta D_{i-1,i})$ , where  $\mathcal{F}(\cdot)$  denotes a RNN unit (e.g., vanilla RNN, LSTM or GRU), and  $h_i$  denotes the hidden state encoding the historical information of the sequence up to the time step i. In the current literature, this scheme has been instantiated by either using spatiotemporal-specific transition matrices parameterized by  $\Delta T_{i-1,i}$  and  $\Delta D_{i-1,i}$  in RNNs [7, 30], or extending/adding gates controlled by  $\Delta T_{i-1,i}$  and  $\Delta D_{i-1,i}$  to LSTM [22, 67, 68]. However, this scheme cannot fully benefit from rich spatiotemporal contexts for handling sparse user mobility trajectories, due to the following reasons.

• From a temporal perspective, feeding temporal distances between successive check-ins to the RNN unit often ignores the *temporal periodicity* of user mobility. Specifically, the periodicity of human activities is universal [16, 28]. Figure 1(a) shows the return probability of user check-ins over time, defined as the probability of a user re-checking in at a POI a certain period of time after her first check-in at that POI, on our collected Foursquare dataset. We observe a clear daily (periodic) revisiting pattern. In the context of location prediction, such a periodicity implies that historical check-ins with a temporal distance (taken from the current time) closer to these daily peaks (1 day, 2 days, etc.) have higher predictive power. However, iteratively feeding temporal distances between *successive* check-ins into the RNN unit often fails to benefit from this periodicity property. Figure 1(a) illustrates such an example, where  $\Delta T_{i-1,i}$  and  $\Delta T_{i,i+1}$  refers to the temporal distances between successive check-in pairs  $p_{i-1} \rightarrow p_i$  and  $p_i \rightarrow p_{i+1}$ , respectively. If we consider the temporal distance in two steps





(a) Temporal Periodicity (

(b) Spatial Regularity

Fig. 1. Spatiotemporal factors in user check-in data. a) Temporal factor shown as periodicity, where the example in the box shows that considering temporal distances between successive check-ins only ( $\Delta T_{i-1,i}$  and  $\Delta T_{i,i+1}$ ) cannot capture such a periodicity. b) Spatial factor where the example shows that considering spatial distances between *successive* check-ins only ( $\Delta D_{i-1,i}$  and  $\Delta D_{i,i+1}$ ) cannot capture the proper distances  $\Delta D_{i-1,i+1}$ . In both cases, the corresponding techniques fail to fully benefit from the historical check-ins with high predictive power when predicting location.

 $(p_{i-1} \to p_{i+1})$ , we find that  $\Delta T_{i-1,i+1}$  is close to the 1 day peak, indicating that  $p_{i-1}$  is very helpful for predicting the next location. In contrast, this cannot be captured if we iteratively feed temporal distances  $\Delta T_{i-1,i}$  and  $\Delta T_{i,i+1}$  into the RNN unit.

• From a spatial perspective, feeding spatial distances between successive check-ins to the RNN unit oversimplifies the *spatial regularity* of user mobility. Specifically, it has been found that a user's check-ins in a region that she frequently visited are highly biased to certain POIs [7, 62]. In other words, those regions often have certain implicit "functions", such as working or shopping. Subsequently, the closer the user is to such a region, the more predictable her behavior is. This suggests that the closer a past check-in is located to the current location, the more helpful it is for the next location prediction. However, only considering spatial distances between successive check-ins fails to capture such distances over space. Figure 1(b) shows an example. We observe that the spatial distance in two steps  $\Delta D_{i-1,i+1}$  is much smaller than both  $\Delta D_{i-1,i}$  and  $\Delta D_{i,i+1}$  (which is also evidenced by our empirical analysis on real-world datasets in Figure 5), suggesting that  $p_{i-1}$  is more helpful for location prediction. In contrast, this cannot be captured if we consider spatial distances  $\Delta D_{i-1,i}$  and  $\Delta D_{i,i+1}$  only.

Against this background, we propose in this paper Flashback++, a general RNN architecture designed for modeling sparse user mobility trajectories, leveraging rich spatiotemporal contexts to robustly overcome the sparsity issue of user mobility trajectories by learning to flash back in hidden states of RNNs. More precisely, departing from the widely adopted scheme of adding spatiotemporal factors into the recurrent hidden state passing process of the RNNs, our method explicitly uses the spatiotemporal context to automatically search past hidden states with high predictive power (i.e., historical hidden states that share similar spatiotemporal contexts to the current one). To fully benefit from the rich spatiotemporal contexts, Flashback++ is guided by the two universal mobility laws, i.e., temporal periodicity and spatial regularity as shown in Figure 1(a) and 1(b), respectively, and optimally integrate them into the RNN architecture by learning to re-weighting the past hidden states for next location prediction. Moreover, as we do not modify the hidden state passing process of RNNs (while many existing techniques do), our Flashback++ can be easily instantiated with any RNN units (e.g., vanilla RNN, LSTM or GRU). Our contributions are summarized as follows:

111:4 Deng et al.

By revisiting existing RNN-based location prediction techniques, we reveal that the widely
adopted scheme of adding spatiotemporal factors into RNNs mismatches the universal spatiotemporal mobility laws.

- We propose Flashback++, a general RNN architecture designed for modeling sparse user
  mobility trajectories. It not only leverages rich spatiotemporal contexts to search past hidden
  states with high predictive power, but also learns to optimally combine them via a hidden
  state re-weighting mechanism, which significantly improves the robustness of the models
  against different settings and datasets.
- We conduct a thorough evaluation of our method compared to a sizable collection of baselines
  on two real-world LBSN datasets and one on-campus mobility dataset. Results show that
  Flashback++ consistently and significantly outperforms all baseline techniques, yielding an
  improvement of 20.56% to 44.36% over the best-performing spatiotemporal sequence models.

Compared to our previous work Flashback [56] with non-learnable spatiotemporal decay parameters, Flashback++ further learns these parameters to optimally integrate the two universal mobility laws, and achieves significant improvements of 10.18% to 20.15% in next location prediction tasks. Moreover, compared to Flashback, Flashback++ can automatically learn these parameters to adapt to different model settings (different RNN architectures and numbers of hidden states to flash back), different levels of trajectory sparsity, and different train-testing splitting ratios, which significantly improves its robustness by 31.05% to 94.60% against different cases. In summary, beyond the previous idea of "flash back in the hidden states", this extension makes the model learn to "flash back to the right moment".

# 2 RELATED WORKS

## 2.1 Human Mobility Trajectory

The study on human mobility can be dated from 1885, when Ravenstein published his work on studying migration using human mobility trajectories extracted from demographic data [42]. Nowadays, sensor-embedded smart devices make human mobility trajectory data more accessible for large-scale studies [8]. According to the data collection scheme, human mobility trajectory data can be classified into two categories as follows.

First, continuously sampled trajectories contain sequences of locations regularly collected from devices carried by users. For example, GeoLife project [69] conducted by Microsoft Research Asia collected mobility trajectories of 182 users in a period of over three years (from April 2007 to August 2012) by recording a GPS coordinates every 5 seconds or every 10 meters; the Lausanne Data Collection Campaign [24] conducted by Nokia Research Center, IDIAP Research Institute, and EPFL from 2009 to 2011, collected the mobility trajectories of 200 participants; each participant possesses a smart phone programmed to record various sensor readings (e.g., WLAN, GPS, Bluetooth) periodically (e.g., every 60 seconds). Although such continuously sampled mobility trajectory data contains fine-grained user mobility trajectories, they often have very limited scales (a small number of individuals) due to the controlled experimental settings and participants' privacy concerns (on installing background monitoring software on personal devices, for example).

Second, *voluntarily shared* trajectories contain sequences of self-reported locations (mostly by users on social media platforms). For example, on LBSNs, millions of users voluntarily share their check-in activity with their friends online, resulting in a large-scale collection of user mobility trajectories. However, users are often willing to share only part of their activities by checking-in at POIs; in other words, a user might visit some POIs without checking-in there, due to several reasons such as uninteresting places, privacy concerns, or simply forgetting to do it. Therefore, although LBSNs have been widely recognized as a primary data source for large-scale human

mobility studies, the social sharing basis intrinsically makes the resulting mobility trajectory *sparse* [17, 46, 59]. The resulting trajectories can only be regarded as a social representation of users' daily activities, as the users often have different preferences on what kind of activities to share, which intrinsically differs from the continuously sampled trajectories. Such sparsity needs to be carefully considered when performing data analytic tasks [51]. Against this background, we investigate in this paper the location prediction problem over sparse trajectories using RNNs.

#### 2.2 Location Prediction

Location prediction is a key problem in human mobility modeling, which predicts the location of a user based on the user's historical mobility trajectories. Traditional methods for location prediction often resort to various mobility features, such as hand-craft features including historical visit counts [15, 39, 57], activity preferences [62, 64], social relationships [5, 45], or automatically-learnt features using graph embedding techniques [12, 40, 55, 59, 60]. In addition, generative/factorization models have also been used to solve location prediction/recommendation problems [23, 29, 52, 61, 65]. However, these techniques have limited capability of capturing the sequential patterns of user mobility.

To capture user sequential mobility patterns, (Hidden) Markov Chains have been widely used for sequential prediction [4, 13, 36]. The basic idea is to estimate a transition matrix encoding the probability of a behavior based on previous behaviors. A typical technique here is Factorizing Personalized Markov Chains (FPMC) [44], which estimates a personalized transition matrix via matrix factorization techniques. FPMC has been extended to the location prediction problem by further considering spatial constraints [4, 13] in building the transition matrices. However, one drawback of these location prediction techniques lies in the fact that they fail to well capture the long-term dependency on user mobility trajectories.

Recently, Recurrent Neural Networks (RNNs) have been shown as a successful tool to model sequential data [37], capturing complex long- and short-term dependency over input sequences. To handle sparse and incomplete sequences, existing techniques strive to add context factors into the RNNs. For example, temporal factors can be added by truncating each sparse input sequence into several short sessions [11, 19], or by considering temporal factors as additional inputs to the RNN units [38, 70]. For the problem of location prediction over sparse user mobility sequences, spatiotemporal factors have been shown as strong predictors [62]. The most popular scheme to incorporate spatiotemporal factors into RNNs is adding the spatiotemporal distances between (mostly successive) check-ins as additional inputs to the RNN units. For example, Distance2Pre [7] uses the distance between successively checked POIs as additional input to RNNs; STRNN [30] uses spatiotemporal-specific transition matrices parameterized by the spatiotemporal distances in RNNs; HST-LSTM [22] extends existing gates in LSTMs to let these gates take the spatiotemporal distance as an additional input; STGN [68] adds additional gates controlled by the spatiotemporal distances to LSTMs; NeuNext [67] further extend STGN leveraging POI context prediction to assist next POI recommendation tasks by joint learning. However, as discussed in the Introduction, such schemes cannot fully benefit from the rich historical spatiotemporal contexts encoded in mobility trajectories.

Our previous work proposed a flashback mechanism [56] to explicitly use the high-order spatiotemporal distances to search past hidden states with high predictive power. This flashback mechanism has been later adopted by different works. For example, BiGRU [3] designs a bi-directional GRU with the flashback mechanism; BSDA [25] designs a bi-direction speculation method for location prediction, combining flashback with an additional RNN which models POIs' appeal to users; Bi-STAN [54] extends flashback mechanism for missing check-in imputation; RTPM [31]

111:6 Deng et al.

combines flashback with real-time user preference models for real-time location recommendation; Graph-Flashback [41] incorporates User-POI knowledge graph with user social relationships into the flashback framework. However, these works all use the hand-craft spatiotemporal decay parameters as suggested by Flashback. In this paper, we extend Flashback to Flashback++ with learnable spatiotemporal decay parameters, and conducted extensive experiments showing that this extension not only achieves large improvements of 10.18% to 20.15% in next location prediction tasks, but also significantly improves the model robustness by 31.05% to 94.60% against different settings (different RNN architectures and numbers of hidden states to flash back) and different levels of trajectory sparsity.

Note that other sequence models (apart from RNNs) have also been used for location prediction tasks. Some models use spatiotemporal attention networks. For example, STAN [34] uses relative spatiotemporal information between non-adjacent check-ins and DeepMove [11] combines an attention layer with gated recurrent units (GRU) to learn long-term periodicity and short-term sequential patterns. Others use graph-enhanced attention networks. For example, SGRec [26] aggregates information from both neighboring POI nodes and edges to expressively embed POIs in a sequence to learn sequential dependencies from sparse POI-level interactions; GSTN [53] learns distance-based and transition-based geographical latent representations via on graph embeddings to capture high-order complex geographical influences among POIs; GetNext [63] uses graph convolutional networks on the trajectory flow map to obtain a POI-to-POI probability map and reinforces the representations of POIs; GeoSAN [27] proposes a self-attention based geography encoder to represent the exact GPS positions of locations in order to capture long-term sequential dependence and thus effectively make use of geographical information. Our work differs from them in the aspect that instead of letting the model freely search and utilize historical hidden states for example, our method Flashback++ designs an intuitive and principled mechanism letting the model learn to search and utilize historical hidden states under the constraints of the two spatiotemporal mobility laws, which shows superior performance as evidenced in our experiments later.

## 3 FLASHBACK++

Flashback++ is designed for modeling sparse user mobility trajectories, with a particular focus on leveraging rich spatiotemporal contexts by doing flashbacks on hidden states in RNNs. Instead of implicitly considering context factors by adding spatiotemporal factors into the recurrent hidden state passing process of RNNs (as most existing techniques do), our solution explicitly uses the spatiotemporal contexts to search past hidden states with high predictive power (i.e., historical hidden states that share similar contexts as the current one). To fully benefit from the rich spatiotemporal contexts, Flashback++ optimally integrates the two universal mobility laws, i.e., temporal periodicity and spatial regularity, into the RNN architecture by learning to re-weighting the past hidden states for next location prediction.

## 3.1 Overview

Figure 2 shows an overview of our Flashback++ architecture. Given a user's mobility trajectory represented as a POI sequence  $\{..., p_{i-3}, p_{i-2}, p_{i-1}, p_i, p_{i+1}...\}$ , we denote the temporal and spatial distances between two check-ins  $p_i$  and  $p_j$  as  $\Delta T_{i,j}$  and  $\Delta D_{i,j}$ , respectively. As shown in Figure 2, our recurrent hidden state passing process remains unaltered from classical RNNs, i.e.,  $h_i = \mathcal{F}(p_i, h_{i-1})$ , letting RNNs capture sequential user mobility patterns. However, instead of using only the current hidden state  $h_i$  to predict the next location  $p_{i+1}$  (as classical RNNs do), we leverage the spatiotemporal context to search past hidden states with high predictive power. To this end, we compute the weighted average of the historical hidden states  $h_j$ , j < i, with a learnable weight  $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$  as an aggregated hidden state. This weight integrates on one hand the two universal mobility

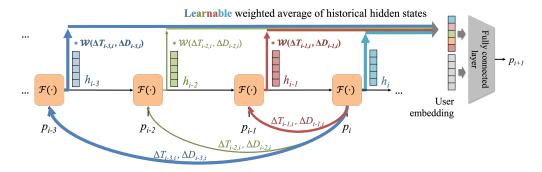


Fig. 2. Overview of our Flashback++ architecture for next location prediction

laws, i.e., temporal periodicity and spatial regularity, which are parameterized by the temporal and spatial distances ( $\Delta T_{i,j}$  and  $\Delta D_{i,j}$ , respectively) between check-in  $p_i$  and  $p_j$ ; on the other hand, the weight also incorporates two learnable parameters  $\alpha$  and  $\beta$ , allowing the model to search for (under the two universal mobility laws) the most predictive hidden states across time and space, respectively, for accurate location prediction (more information on this point below). Finally, to model individual user's preferences, we define a learnable user embedding vector for each user, which is concatenated with the aggregated hidden state and then fed into a fully connected layer for predicting the next location, as shown in Figure 2.

In summary, to effectively predict locations from sparse user mobility trajectories, Flashback++ 1) uses RNNs to capture *sequential* patterns, 2) leverages *spatiotemporal* contexts to automatically search and learn to combine past hidden states with high predictive power, and 3) incorporates user embeddings to consider users *preferences*.

# 3.2 Context-Aware Hidden State Weighting

The weight  $W(\Delta T_{i,j}, \Delta D_{i,j})$  is designed to measure the predictive power of the hidden state  $h_j$  according to its spatiotemporal contexts.

First, from a temporal perspective, our primary goal is to incorporate the temporal periodicity property of user behavior (as shown in Figure 1(a)) into  $W(\Delta T_{i,j}, \Delta D_{i,j})$ . To this end, we resort to a Havercosine function, a typical periodic function with outputs bounded in [0, 1], parameterized by  $\Delta T_{i,j}$  (in days) as follows:

$$w_{period}(\Delta T_{i,j}) = \text{hvc}(2\pi \Delta T_{i,j}) \tag{1}$$

where  $\text{hvc}(x) = \frac{1+\cos(x)}{2}$  is the Havercosine function modeling the daily periodicity. Moreover, as we can see from Figure 1(a), the return probability exponentially decreases when increasing  $\Delta T_{i,j}$ , which indicates that besides the periodicity, the older a check-in is, the less impact it has for prediction. Subsequently, we add a temporal exponential decay weight to model this factor:

$$w_T(\Delta T_{i,j}) = w_{period}(\Delta T_{i,j}) \cdot e^{-\alpha \Delta T_{i,j}}$$
  
= hvc(2\pi \Delta T\_{i,j}) \cdot e^{-\alpha \Delta T\_{i,j}} (2)

where  $\alpha$  is a temporal decay rate, controlling how fast the weight decreases over time  $\Delta T_{i,j}$ .

Second, from a spatial perspective, we consider the spatial regularity of user behavior (as shown in Figure 1(b)); it suggests that the closer a check-in is to the current location, the more helpful it is for location prediction. Accordingly, we use a distance exponential decay weight to model this factor:

$$w_S(\Delta D_{i,j}) = e^{-\beta \Delta D_{i,j}} \tag{3}$$

111:8 Deng et al.

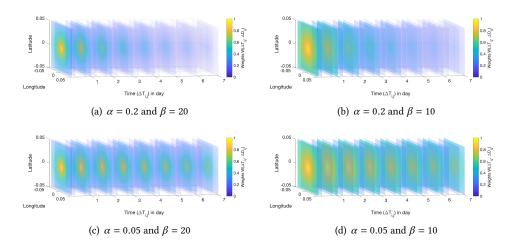


Fig. 3. Visualization of the spatiotemporal context-aware weight  $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$  over space and time for different combinations of values for  $\alpha$  and  $\beta$ .  $\Delta D_{i,j}$  is illustrated as L2 distance from the origin over spatial space (latitude and longitude axes). The transparency of the slices on the time axis is proportional to the weights (the lower the weight is, the more transparent a slice is).

where  $\Delta D_{i,j}$  is the L2 distance between the GPS coordinates of POIs  $p_i$  and  $p_j$ , and  $\beta$  is a spatial decay rate, controlling how fast the weight decreases over spatial distance  $\Delta D_{i,j}$ .

Finally, we obtain the weight  $W(\Delta T_{i,j}, \Delta D_{i,j})$  by combining the temporal and spatial weights together:

$$W(\Delta T_{i,j}, \Delta D_{i,j}) = w_T(\Delta T_{i,j}) \cdot w_S(\Delta D_{i,j})$$
  
= hvc( $2\pi \Delta T_{i,j}$ ) $e^{-\alpha \Delta T_{i,j}}e^{-\beta \Delta D_{i,j}}$  (4)

where the first Havercosine term captures the periodicity property of user check-ins, and the exponential terms model the spatiotemporal decay of the impact of historical check-ins on location prediction. In essence, the context-aware weight  $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$  integrates the two universal mobility laws to measure the predictive power of hidden states.

# 3.3 Learning to Re-Weight Past Hidden States

The context-aware weight in Eq. 4 incorporates two parameters  $\alpha$  and  $\beta$  to control where to flash back temporally and spatially, respectively, under the two mobility laws via weight decaying on past hidden states. Figure 3 shows a visualization of the weights over space and time for different values of  $\alpha$  and  $\beta$ . We observe on one hand a periodicity pattern over time ( $\Delta T_{i,j}$ ), and on the other hand, a spatiotemporal decay over space and time. Comparing the cases across different values of  $\alpha$  and  $\beta$ , we observe that lower values of  $\alpha$  and  $\beta$  spreads the weight further over space and time. Subsequently, tuning  $\alpha$  and  $\beta$  gives the flexibility to control when and where should we flash back on hidden states, under the constraints of the two universal mobility laws.

In this paper, different from our previous work [56] where we manually tune  $\alpha$  and  $\beta$ , Flashback++ introduces both of them as learnable parameters that are trained together with the RNNs over epochs. This design further provides the model with the flexibility of learning to optimally integrate the two universal mobility laws into the RNN architectures for next location prediction, i.e., *learning to flash back to the right moment* under the constraints of the two mobility laws. This significantly improves the robustness of Flashback++, in particular for the following three cases.

- First, when RNN architectures (e.g., RNN, LSTM, or GRU) have different modeling capacities,  $\alpha$  and  $\beta$  can be learnt to adapt to them for achieving optimal location prediction performance. For example, if the current hidden state encodes insufficient information from the trajectory for predicting the next location,  $\alpha$  and  $\beta$  will be learnt to have low values such as in Figure 3(d), resorting to more information from historical hidden states for prediction.
- Second, when the number of available hidden states (to be combined via weighted average) changes,  $\alpha$  and  $\beta$  can be learnt to optimally use these hidden states. For example, when more hidden states are available,  $\alpha$  and  $\beta$  tend to be learnt to have lower values so as to utilize those "older" hidden states (by assigning higher weights to them) for boosting the performance of location prediction.
- Third, when learning from mobility trajectory datasets with different levels of sparsity,  $\alpha$  and  $\beta$  can be learnt to utilize the right amount of information from historical hidden states for prediction. For example, for a more sparse trajectory dataset,  $\alpha$  and  $\beta$  are learnt to have low values such as in Figure 3(d), using more information from historical hidden states for prediction.

All these cases are evidenced by our experiments below.

#### 4 DISCUSSIONS

## 4.1 Why Does it Work?

By flashing back to the historical hidden states, our method can discount the "noise" from the recurrent hidden state passing process of the RNNs over sparse user mobility trajectories, and also create an explicit "attention" mechanism (more discussion on this point below) by leveraging past hidden states with high predictive power (i.e., historical hidden states that share similar contexts as the current hidden state) for location prediction. Figure 4 shows a toy example from the temporal perspective. On one hand, Figure 4(a) shows an actual (complete) user mobility trajectory with a clear sequential pattern ("Home-Office-Restaurant-Office-Shopping-Bar-Home"), where classical RNNs can effectively capture such a pattern and predict the next location "Home". On the other hand, for an observed (sparse) user mobility trajectory as shown in Figure 4(b), the sequential pattern is difficult to be captured by RNNs, where the hidden state passing process becomes noisy due to the incompleteness of the sequence. Even when considering temporal distances between successive check-ins as additional inputs of the RNN units (as many state-of-the-art techniques do), it still falls short in capturing long-term temporal (i.e, periodicity) dependencies. However, by flashing back to the historical hidden states sharing a similar temporal context as the current one ( $\Delta T \approx 1$  day capturing the daily periodicity, i.e., at a similar time on the previous day), we can predict the next location "Home".

### 4.2 Relation to Attention Mechanisms

The attention mechanism [2] was initially introduced to improve the performance of the encoder-decoder model for machine translation using RNNs. Its key idea is to give the decoder more flexibility to automatically search and utilize the most relevant tokens of the input sequence, by learning a weighted combination of all the encoded input tokens, where higher weights are associated with more relevant tokens. Specifically, in a sequence-to-sequence (seq2seq) task such as machine translation, instead of encoding the input sequence into one hidden state vector<sup>1</sup> for decoding, the attention model computes a weighted combination of historical hidden states specifically for each output step. Our Flashback++ is related to the attention mechanism in the following aspects. On

 $<sup>^{1}</sup>$ Also known as context vector in seq2seq tasks. We keep using hidden state vectors here for the purpose of terminology consistency.

111:10 Deng et al.

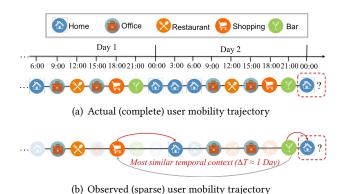


Fig. 4. A toy example illustrating the working principle of Flashback++ from a temporal perspective.

one hand, both of them try to make better use of the input sequence for prediction by leveraging a weighted combination of the historical hidden states with learnable weights. On the other hand, different from the attention mechanism that computes the weights based on the similarity between each input token and the current output token in the learnable word embedding space, Flashback++ computes the weights according to the similarity between the spatiotemporal *context* of each token (each historical check-in) and the current *context*, where the similarity computation further integrates the physical human mobility laws with learnable decay parameters which allow our model to learn to "flash back to the right moment" under the constraints of the two mobility laws for predicting next location.

Moreover, compared to a state-of-the-art technique Deepmove [11] that designs a sophisticated attention mechanism in GRU letting the model *freely* search and utilize historical hidden states, our method Flashback++ designs an intuitive and general mechanism letting the model learn to search and utilize historical hidden states *under the constraints of the two spatiotemporal mobility laws*. Our experiments show that Deepmove not only underperforms Flashback++ in location prediction tasks, but also suffers from an efficiency bottleneck when learning from large scale datasets due to the expensive computation incurred by the attention module (see Table 2 for more detail).

## 4.3 How Far to Flash Back?

As Flashback++ generates an aggregated hidden state from past hidden states, an immediate question here is how many past hidden states should be considered? To answer this question, we review the temporal and spatial exponential decay terms ( $e^{-\alpha\Delta T_{i,j}}$  and  $e^{-\beta\Delta D_{i,j}}$ ) in  $\mathcal{W}(\Delta T_{i,j}, \Delta D_{i,j})$ . Figure 5 shows the box plot of temporal and spatial distances ( $\Delta T$  and  $\Delta D$ , respectively) between two check-ins k-step distant from each other in user trajectories on our LBSN datasets. We observe that both  $\Delta T$  and  $\Delta D$  increase with k; while  $\Delta T$  (almost) linearly increases,  $\Delta D$  quickly flattens out. Taking k=19 on Gowalla as an example (where 20 historical hidden states are considered), median  $\Delta T$  is 13.13 days (315 hours in Figure 5) and median  $\Delta D$  is 0.073 (Euclidean distance between two GPS coordinates<sup>2</sup>, corresponding to 8.15 km in Figure 5). The learnt optimal temporal and spatial decay rates using RNNs are  $\alpha=0.40$  and  $\beta=35.28$ , respectively. Subsequently, the temporal and spatial exponential decay term  $e^{-\alpha\Delta T_{i,j}}e^{-\beta\Delta D_{i,j}}$  is 0.0004. This term is indeed the upper bound of

<sup>&</sup>lt;sup>2</sup>The exact distance between two GPS coordinates should be computed on the basis of a spherical earth (also considering ellipsoidal effects). However, in practice, Euclidean distance is widely used in spatiotemporal data mining due to its simplicity and computational efficiency.

 $W(\Delta T_{i,j}, \Delta D_{i,j})$  as all other terms in  $W(\Delta T_{i,j}, \Delta D_{i,j})$  are bound to [0,1]. In other words, a hidden state more than 19-step back receives a weight less than 0.0004, which contributes little to the aggregated hidden state. We also study this point in our experiments below in Section 5.5, where we show that with learnable  $\alpha$  and  $\beta$ , Flashback++ can well adapt to different numbers of hidden states to boost location prediction performance, and the selection of the number of hidden states needs also to consider the runtime performance.

# 4.4 The complexity of the Flashback mechanism

The complexity of our Flashback mechanism is discussed as follows. Let n be the length of the sequence and d be the size of hidden states (embedding size), respectively. Take vanilla RNNs as an example, the time and space complexities for one sequence are  $O(nd^2)$  and O(nd), respectively. Built on top of RNNs, our flashback mechanism additionally re-weights and combines n hidden states of dimension d, resulting in the time complexity  $O(nd^2 + nd)$  and while space complexity remains the same as O(nd). Therefore, we see that the computational overhead of our Flashback mechanism is very small as hidden state size d is often set to  $d \gg 1$  (e.g., d = 10 in our experiments), and thus  $nd^2 \gg nd$ .

# 4.5 Applicability of the Flashback mechanism to other models

State-of-the-art location prediction models often resort to attention-based models [11] and graph-based models [41]. Our Flashback mechanism can also be integrated with these models. First, the graph-based models often built graphs of users (social networks) or POIs (transition networks) to explicitly capture their high-order correlations for improved performance. Our Flashback mechanism is applied to temporal sequences; it is orthogonal to the graph-based approaches and thus can be easily integrated with these approaches. For example, Graph-Flashback [41] integrates our Flashback mechanism with POI transition graph models. Second, the attention-based mechanism learns to re-weigh and combine all historical hidden states freely, while our Flashback mechanism constrains the re-weighing and combination process under the two mobility laws. In this context, our Flashback can also be integrated with the attention mechanism by designing a more flexibly learnable re-weighting process guided by the two mobility laws. This remains our future work.

# **5 EXPERIMENTS**

# 5.1 Experimental Setup

5.1.1 Dataset. We conduct experiments using three real-world datasets.

First, we use two widely used check-in datasets collected from two LBSNs **Gowalla** [5] and **Foursquare** [56], respectively. A user's mobility trajectory is represented as a sequence of check-ins at POIs. Due to the social sharing nature of LBSNs, such user mobility trajectories are intrinsically sparse. We use these two datasets to evaluate the performance of location prediction by comparing different techniques.

Second, we collect an in-house Wi-Fi-positioning-based mobility dataset on the campus of the University of Macau (**UM-WiFi**). Specifically, a user's mobility trajectory is represented as a sequence of the (automatic) connection records between the user's mobile devices and Wi-Fi Access Points (APs). Each connection record includes an authenticated user ID (hashed for the purpose of privacy protection), a connection time stamp, and the ID of the connected AP; it can be regarded as an *automatic* check-in of the user at the AP without human intervention<sup>3</sup>. On the campus of the University of Macau, about 6,000 APs have been deployed, covering over 80% of

<sup>&</sup>lt;sup>3</sup>Although these automatic device-to-device check-ins at APs intrinsically differ from the user voluntarily shared check-ins at POIs on LBSNs, for the sake of brevity, we keep using "check-ins" and "POIs" for both cases.

111:12 Deng et al.

	- "	_	
Dataset	Gowalla	Foursquare	UM-WiFi
#Users	52,979	46,065	11,155
#POIs	121,851	69,005	5,920
#Checkins	3,300,986	9,450,342	48,370,506
Collection	02/2009~	04/2012~	01/2021~
period	10/2010	01/2014	04/2021
Median time between	11.09 hours	16.72 hours	0.03 hours
successive check-ins	(0.46 days)	(0.70  days)	0.05 nours

Table 1. Statistics of the datasets

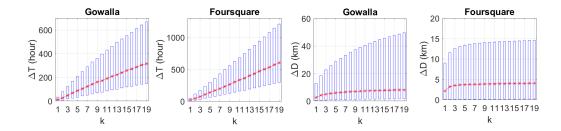


Fig. 5. Box plot of temporal and spatial distances ( $\Delta T$  and  $\Delta D$ , respectively) between two check-ins k-step distant in check-in sequences on Gowalla and Foursquare datasets.

the campus (including both indoor and outdoor areas), providing Internet services to over 10,000 students and staff. Subsequently, with the automatic nature of these device-to-device check-ins and the high coverage of the campus by the APs, this dataset contains comprehensive mobility traces on campus, which is much more dense compared to the two LBSN datasets. We use this dataset to investigate the robustness of our proposed method against trajectories of different levels of sparsity in a controlled experiment setting by sparsifying the (dense) trajectories via random sampling to different extents. Such a setting ensures the trajectories of different levels of sparsity have the same underlying mobility pattern, thus supporting a fair comparison between them. Note that these artificially-sparsified datasets still differ from the LBSN datasets. The latter can be regarded as manually-sparsified datasets by users, which strongly involves a user's personal preference on sharing what kind of check-ins from all her daily activities.

Table 1 shows the statistics of all three datasets. We observe that the two LBSN datasets are much more sparse (with median time between successive check-ins being 11.09 and 16.72 hours, respectively), compared to the UM-WiFi dataset with a median time of 0.03 hours. Figure 5 further shows the box plot of temporal and spatial distances ( $\Delta T$  and  $\Delta D$ , respectively) between two checkins k-step distant from each other in user trajectories on Gowalla and Foursquare. We observe that both  $\Delta T$  and  $\Delta D$  (and their variances) increase with k. While  $\Delta T$  (almost) linearly increases,  $\Delta D$  quickly flattens out. This further verifies our motivation of capturing spatial regularity as shown in Figure 1(b), where considering spatial distances between *successive* check-ins only ( $\Delta D_{i-1,i}$  and  $\Delta D_{i,i+1}$ ) cannot capture the proper distance  $\Delta D_{i-1,i+1}$ .

- *5.1.2 Baselines.* We compare Flashback++ against a sizable collection of state-of-the-art techniques from five categories:
- User Preference-based Methods. WRMF [21] is a matrix factorization based recommendation technique designed explicitly for implicit feedback such as historical user behavior, where users do

- not explicitly express their preference; it fits well our check-in datasets that contain historical user presence at POIs. **BPR** [43] is a ranking-based recommendation technique for implicit feedback, which learns user preferences by minimizing a pairwise ranking loss; it uses a bootstrap sampling technique to learn from both positive feedback and non-observed feedback to alleviate data sparsity issue. We use the default parameter settings of LibRec<sup>4</sup> for these two methods.
- Feature-based methods: Most Frequent Time (MFT, the best-performing feature by [15]) ranks a POI according to a user's historical check-in count at a POI and at a specific time slot in the training dataset; we define the timeslot as 24 hours in a typical day as suggested by [62]. LBSN2Vec [59] is an automatic feature learning technique designed specifically for check-in data; it learns user, time and POI feature vectors from an LBSN hypergraph, and ranks a POI according to its similarities with user and time in the feature space for location prediction tasks; we set the number of negative samples, random walk window size, and the mobility data ratio to {10, 10, 1} in our experiments.
- Markov-Chain-based Methods: FPMC [44] estimates a personalized transition matrix of POIs in user mobility trajectories using matrix factorization techniques. We set the number of negative samples and the dimension of factorization to {10, 32}. PRME [13] extends FPMC to location prediction problems by further considering spatial constraints, and learns user and POI embeddings to capture the personalized POI transition patterns. We set the threshold, component weight, and hidden state size to {360, 0.2, 20}. TribeFlow [14] learns a mixture model to capture the semi-Markov transition probability matrix over latent environments for predicting user trajectories. We set the number of cores, number of transitions, number of iterations, and the number of batches to {20, 0.3, 2000, 20}, and use ECCDF-based kernel introduced in [14].
- Basic RNNs: RNN [66] is a vanilla Recurrent Neural Network architecture, capturing sequential
  patterns from user mobility trajectories for location prediction. LSTM (Long Short-Term Memory)
  [20] is capable of learning from both short- and long-term dependency in sequences using a
  memory cell controlled by three multiplicative gates including an input gate, an output gate and
  a forget gate. GRU (Gated Recurrent Unit) [6] captures long-term dependency by controlling
  information flow with an update gate and a reset gate. We set the hidden state size to 10 for all
  three RNN architectures and all the datasets.
- Spatiotemporal Sequence Models: **DeepMove** [10, 11] adds an attention mechanism to GRU for location prediction over sparse mobility trajectories. We set the size of all embeddings and the hidden state size to {50, 10}, and use average history mode and LSTM as suggested by [11]. STRNN [30] uses customized transition matrices parameterized by the spatiotemporal distances between check-ins within a time window in RNNs. STGN [68] add additional gates controlled by the spatiotemporal distances between successive check-ins to LSTM. STGCN [68] is an extension of STGN with coupled input and forget gates for improved efficiency. We set the hidden state size to 10 for all the datasets. STAN [34] designs a spatiotemporal attention network capturing interactions between non-adjacent locations and non-consecutive check-ins using spatiotemporal contexts. We set the embedding size, the maximum length for trajectory sequence, and the number of negative samples to {50, 100, 10}, respectively. GetNext [63] exploits the user flow trajectory graph and time-aware category context embedding for next location prediction; for a fair comparison, we do not use category information by assigning the same category to all POIs. We set the embedding size of POIs and users, the numbers of three GCN layers' channels, the embedding size of the transformer encoder to {128, 32, 64, 128, 1024}, respectively. Flashback [56] is our previous work where our context-aware hidden state weighting mechanism is also used, but with non-learnable spatiotemporal decay parameters  $\alpha$  and  $\beta$ ; we instantiate it using

<sup>&</sup>lt;sup>4</sup>https://github.com/guoguibing/librec

111:14 Deng et al.

the three basic RNNs, named as **Flashback (RNN)**, **Flashback (LSTM)**, and **Flashback (GRU)**, and set  $\alpha$  and  $\beta$  as suggested in [56]. We set the hidden state size to 10 of the three Flashback RNN-based architectures for all the datasets.

Note that for a fair comparison, we exclude the methods for the next location prediction using additional information beyond historical user mobility traces from our baselines. For example, Graph-Flashback [41] uses user social networks as additional data; Bi-STAN [54], BSDA [25], RTPM [31] predict locations conditioned on a "query" time (or time interval), i.e., where a user will go at a given time in the future, which differs from our problem setting.

For our proposed Flashback++, we also instantiate it using the three basic RNNs, named as **Flashback++** (RNN), **Flashback++** (LSTM), and **Flashback++** (GRU). We set the hidden state size to 10 for all the datasets, for a fair comparison. We train Flashback++ by backpropagation through time using the Adam stochastic optimizer with cross-entropy loss. The implementation of Flashback++ and LBSN datasets are available here<sup>5</sup>.

5.1.3 Evaluation Protocol and Metrics. We evaluate Flashback++ in the next location prediction task, where we predict where a user will go next, given a sequence of her historical check-ins, as shown in Figure 2. We chronologically split all the mobility trajectories into 80% for training and 20% for test. We report two widely used metrics for location prediction: average Accuracy@N (Acc@N), where N = 1, 5, 10, and Mean Reciprocal Rank (MRR).

# 5.2 Location Prediction Performance Comparison

Table 2 shows the results of comparing different methods on both Gowalla and Foursquare datasets. In general, we observe that Flashback/Flashback++ consistently and significantly outperforms all baseline techniques. In particular, compared to the best-performing baselines (spatiotemporal sequence models in most cases), Flashback++ shows an improvement of 44.36% and 20.56% in MRR, on Gowalla and Foursquare, respectively. Compared to the basic RNNs, our Flashback++ consistently yields significant improvements of 84.20% and 121.35% on Gowalla and Foursquare, respectively, showing the effectiveness of leveraging past hidden states for next location prediction. Compared to our previous work Flashback with *non-learnable* spatiotemporal decay parameters, Flashback++ also achieves consistent improvements of 20.15% and 10.18% on Gowalla and Foursquare, respectively; this result shows the advantage of learning spatiotemporal decay parameters  $\alpha$  and  $\beta$ , giving flexibility of learning to optimally integrate the two universal mobility laws into the RNN architectures for next location prediction. In the following, we experimentally show that these learnable parameters also significantly improve the robustness of the model against different settings (different RNN architectures and numbers of hidden states to flash back) and different levels of trajectory sparsity.

# 5.3 Ablation Study

We conduct an ablation study on our proposed method Flashback++, considering the following variants.

- **Flashback++** (w/o learnable  $\alpha$ ) is a variant of Flashback++ without the learnable temporal decay rate  $\alpha$ . It is also equivalent to Flashback by making the spatial decay rate  $\beta$  learnable.
- Flashback++ (w/o learnable  $\beta$ ) is a variant of Flashback++ without the learnable temporal decay rate  $\beta$ . It is also equivalent to Flashback by making the temporal decay rate  $\alpha$  learnable.
- **Flaskback** is also considered as a variant of Flashback++ without the learnable spatiotemporal decay rates  $\alpha$  and  $\beta$ .

 $<sup>^5</sup> https://github.com/Pursue1221/FlashbackPlusPlus\\$ 

Table 2. Location prediction performance comparison on both Gowalla and Foursquare datasets. The best-performing baselines and Flashback/Flashback++ are highlighted. (\*Experiments of DeepMove are conducted on 5,000 randomly sampled users, due to its poor efficiency where it takes more than 24 hours per epoch using an NVIDIA V100 GPU for all users on both of our datasets; <sup>†</sup>Experiments of STAN are conducted on 2000 users following the setting suggested by [41], due to its poor scalability on large scale datasets. <sup>⋄</sup>Experiments of GetNext are conducted on 15000 randomly sampled users on Foursquare, due to its poor efficiency where it takes more than 24 hours per epoch on the whole Foursquare datasets. )

Method			Gov	walla		Foursquare			
101	letilou	Acc@1	Acc@5	Acc@10	MRR	Acc@1	Acc@5	Acc@10	MRR
User Preference	WRMF	0.0112	0.0260	0.0367	0.0178	0.0278	0.0619	0.0821	0.0427
based Methods	BPR	0.0131	0.0363	0.0539	0.0235	0.0315	0.0828	0.1143	0.0538
Feature-based	MFT	0.0525	0.0948	0.1052	0.0717	0.1945	0.2692	0.2788	0.2285
Methods	LBSN2Vec	0.0864	0.1186	0.1390	0.1032	0.2190	0.3955	0.4621	0.2781
Markov-Chain	FPMC	0.0479	0.1668	0.2411	0.1126	0.0753	0.2384	0.3348	0.1578
based Methods	PRME	0.0740	0.2146	0.2899	0.1503	0.0982	0.3167	0.4064	0.2040
based Methods	TribeFlow	0.0256	0.0723	0.1143	0.0583	0.0297	0.0832	0.1239	0.0645
	RNN	0.0881	0.2140	0.2717	0.1507	0.1824	0.4334	0.5237	0.2984
Basic RNNs	LSTM	0.0621	0.1637	0.2182	0.1144	0.1144	0.2949	0.3761	0.2018
	GRU	0.0528	0.1416	0.1915	0.0993	0.0606	0.1797	0.2574	0.1245
Spatiotemporal	DeepMove*	0.0625	0.1304	0.1594	0.0982	0.2400	0.4319	0.4742	0.3270
Sequence	STRNN	0.0900	0.2120	0.2730	0.1508	0.2290	0.4310	0.5050	0.3248
Models	STGN	0.0624	0.1586	0.2104	0.1125	0.2094	0.4734	0.5470	0.3283
Wiodels	STGCN	0.0546	0.1440	0.1932	0.1017	0.1878	0.4502	0.5329	0.3062
	STAN <sup>†</sup>	0.0891	0.2096	0.2763	0.1523	0.2265	0.4515	0.5310	0.3420
	GetNext <sup>⋄</sup>	0.0912	0.2003	0.2487	0.1484	0.1862	0.4702	0.5763	0.3153
	Flashback (RNN)	0.1158	0.2754	0.3479	0.1925	0.2496	0.5399	0.6236	0.3805
Flashback	Flashback (LSTM)	0.1024	0.2575	0.3317	0.1778	0.2398	0.5169	0.6014	0.3654
	Flashback (GRU)	0.0979	0.2526	0.3267	0.1731	0.2375	0.5154	0.6003	0.3631
	Flashback++ (RNN)	0.1352	0.3107	0.3860	0.2190	0.2775	0.5767	0.6536	0.4123
Flashback++	Flashback++ (LSTM)	0.1316	0.3014	0.3744	0.2128	0.2700	0.5659	0.6415	0.4036
	Flashback++ (GRU)	0.1356	0.3124	0.3890	0.2199	0.2724	0.5679	0.6442	0.4057

Table 3. Ablation study on Flashback++

Method	Gowalla			Foursquare		
	RNN	LSTM	GRU	RNN	LSTM	GRU
Flashback	0.1925	0.1778	0.1731	0.3805	0.3654	0.3631
Flashback++ (w/o learnable $\alpha$ )	0.2125	0.2067	0.2143	0.4050	0.4025	0.4004
Flashback++ (w/o learnable $\beta$ )	0.2100	0.2086	0.2133	0.4076	0.4032	0.4046
Flashback++	0.2190	0.2128	0.2199	0.4123	0.4036	0.4057

Table 3 shows the results. First, we observe that the learnable  $\beta$  can effectively improve the location prediction performance. The improvement is evidenced by the superiority of 1) Flashback++ (w/o learnable  $\alpha$ ) over Flashback, with 16.81% and 8.96% improvement on Gowalla and Foursquare, respectively); and 2) Flashback++ over Flashback++ (w/o learnable  $\beta$ ), with 3.12% and 0.51% improvement on Gowalla and Foursquare, respectively. Second, we observe that the learnable  $\alpha$  can also improve the location prediction performance. The improvement is evidenced by the superiority of 1) Flashback++ (w/o learnable  $\beta$ ) over Flashback, with 16.54% and 9.64% improvement on Gowalla and Foursquare, respectively), and 2) Flashback++ over Flashback++ (w/o learnable  $\alpha$ ), with 2.87%

111:16 Deng et al.

Table 4. coefficient of variation in MRR over different RNN architectures (RNN, LSTM, GRU). A large value indicates a large variation in the performance using different RNN architectures, implying a large variation in their modeling capacities.

Method	Gowalla	Foursquare
Basic RNNs	21.73%	41.84%
Flashback	5.59%	2.55%
Flashback++	1.77%	1.11%

Table 5. Learnt  $\alpha$  and  $\beta$  by Flashback++ using different RNN architectures.

Method	Gov	valla	Foursquare		
Method	α	β	α	β	
Flashback++(RNN)	0.4030	35.2778	0.0096	36.5644	
Flashback++(LSTM)	13.5655	65.3906	0.0226	35.3685	
Flashback++(GRU)	0.3435	58.8516	0.0207	50.4289	

and 1.13% improvement on Gowalla and Foursquare, respectively). Finally, compared to Flashback, Flashback++ integrating both learnable spatiotemporal decay rate  $\alpha$  and  $\beta$  yields a significant improvement of 20.15% and 10.18% on Gowalla and Foursquare, respectively.

# 5.4 Robustness against different RNN architectures

In this experiment, we investigate the robustness of our method against different RNN architectures. Specifically, we measure the robustness using the coefficient of variation [9], defined as the ratio of the standard deviation to the mean of a specific metric over different cases. Taking MRR as an example, it is computed as follows:

$$cv = \frac{\sigma(MRR)}{\mu(MRR)} * 100\% \tag{5}$$

where  $\sigma(MRR)$  and  $\mu(MRR)$  compute the standard deviation and mean of MRR over different cases (i.e., different RNN architectures in this experiment), respectively.

Table 4 computes the coefficient of variation in MRR over different RNN architectures. We observe a large variation in the performance of basic RNN, LSTM, and GRU (with a coefficient of variation of 21.73% and 41.84% in MRR on the two datasets, respectively); this large variation implies the different capacities of different RNN architectures modeling sparse user mobility trajectories. Despite their different modeling capacities, our previous work Flashback implemented with RNN, LSTM, and GRU shows a smaller variation in terms of its performance (with a coefficient of variation of 5.59% and 2.55% in MRR on the two datasets, respectively). Moreover, Flashback++ that learns spatiotemporal decay parameters  $\alpha$  and  $\beta$  to optimally integrate the two universal mobility laws into the RNN architectures can further reduce the performance variation over different RNN architectures, showing a coefficient of variation of 1.77% and 1.11% in MRR on the two datasets, respectively, as shown in Table 4. In summary, Flashback++ significantly improves the robustness against different RNN architectures, significantly outperforming basic RNNs and Flashback by reducing the coefficient of variation in MRR by 94.60% and 62.34% on average, respectively.

To further understand the robustness of Flashback++, we study the spatiotemporal decay parameters  $\alpha$  and  $\beta$  learnt using different RNN architectures. Table 5 shows the results. We observe that the learnt values of  $\alpha$  and  $\beta$  vary not only across different RNN architectures, but also across different datasets. Flashback++ can flexibly adapt to different RNN architectures and datasets by

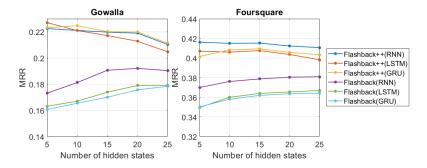


Fig. 6. Location prediction performance over different numbers of hidden states considered

Table 6. coefficient of variation in MRR over different numbers of hidden states considered

Method	Gowalla			Foursquare		
Method	RNN	LSTM	GRU	RNN	LSTM	GRU
Flashback	4.39%	4.14%	4.27%	1.18%	1.92%	1.72%
Flashback++	2.17%	3.89%	2.33%	0.56%	0.94%	0.80%

learning  $\alpha$  and  $\beta$  to maximally boost the performance of location prediction, achieving not only higher performance but also better robustness against different RNN architectures at the same time.

# 5.5 Robustness against different numbers of hidden states to flash back

In this experiment, we investigate the robustness of our method against different numbers of available hidden states to flash back. Specifically, we evaluate the location prediction performance across different numbers of hidden states used by Flashback/Flashback++ on both LBSN datasets. Figure 6 shows the results. First, we see that Flashback++ consistently outperforms Flashback across different numbers of hidden states, by learning spatiotemporal decay parameters  $\alpha$  and  $\beta$  to optimally integrate the two universal mobility laws into the RNN architectures for next location prediction. More importantly, we observe that compared to Flashback, Flashback++ has a smaller variation in performance over different numbers of hidden states in general. Table 6 further computes the coefficient of variation in MRR over different numbers of hidden states. We see that Flashback++ has smaller coefficients of variance than Flashback in all cases. We further investigate the learnt values of  $\alpha$  and  $\beta$  using Flashback++ w.r.t. the number of hidden states considered. As shown in Figure 7, we see that both  $\alpha$  and  $\beta$  decreases when more historical hidden states are considered, which implies that Flashback++ indeed learns to leverage "older" (spatiotemporally more distant) hidden states (by assigning higher weights to them) for boosting the performance of location prediction. In summary, compared to Flashback, Flashback++ not only achieves significant higher performance on location prediction, but also is more robust against the number of hidden states by learning to adapt to different numbers of available hidden states to flash back, reducing the coefficient of variation in MRR by 43.23% on average.

Although Flashback++ yields robust performance on location prediction over different numbers of hidden states, its runtime performance varies. Figure 8 shows the training time per epoch over different numbers of hidden states. We observe that when increasing the number of hidden states, the training time first decreases and then increases; in other words, either too large or small numbers of hidden states lead to longer training time. On one hand, when a smaller number of hidden states are used, the flashing back process takes less time as the sequence length is shorter; however, for

111:18 Deng et al.

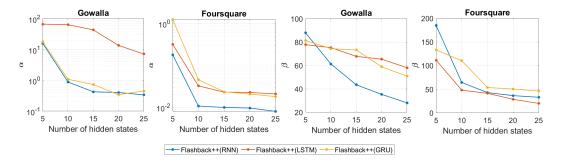


Fig. 7. Learnt  $\alpha$  and  $\beta$  over different numbers of hidden states considered using Flashback++

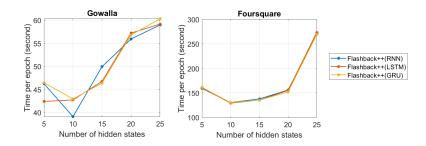
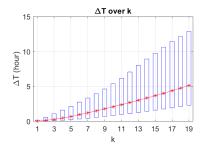


Fig. 8. Runtime performance over different numbers of hidden states considered

the same input trajectories, we have more batches to train, as we split an input trajectory into more sequences (of a shorter length) in the batching process. On the other hand, using a larger number of hidden states requires a longer time to flash back, but with fewer batches (of longer sequences) to train. Subsequently, in general, we observe a U-shaped curve of training time over different numbers of hidden states. In practice, to set the number of hidden states to flash back, we need to consider both the location prediction performance and runtime performance; we suggest 15 or 20 on LBSN datasets.

# 5.6 Robustness against different levels of trajectory sparsity

In this experiment, we evaluate the robustness of our method against trajectories of different levels of sparsity. To this end, we resort to our UM-WiFi dataset, which contains comprehensive and dense trajectories on campus. The left penal of Figure 9 shows the box plot of temporal distances  $\Delta T$  between two check-ins k-step distant in check-in sequences. We see that the UM-WiFi dataset is much more dense (with much smaller  $\Delta T$ ) compared to the two LBSN datasets, and  $\Delta T$  also linearly increases with k. By randomly sampling the original trajectories with different sampling rates, we obtain trajectory datasets with different levels of sparsity. We consider the following five sampling rate 1, 1/2, 1/4, 1/8, and 1/16, where 1 refers to the original (unsampled) dataset and 1/2 refers to randomly sampled 50% check-ins from each trajectory, and so on. The right penal of Figure 9 shows the box plot of temporal distances  $\Delta T$  between successive check-ins in trajectories of different levels of sparsity. We see that  $\Delta T$  exponentially increases with the level of sparsity as the sampling rate exponentially decreases. Using these five datasets of different levels of sparsity, we evaluate the location prediction performance.



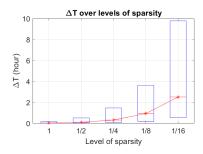


Fig. 9. Box plot of temporal distances  $\Delta T$  on UM-WiFi dataset. The left penal shows  $\Delta T$  between two check-ins k-step distant in check-in sequences. The right panel shows  $\Delta T$  between successive check-ins in trajectories of different levels of sparsity.

Table 7. Performance on UM-WiFi datasets of different levels of sparsity

Method	Level of sparsity					
Method	1	1/2	1/4	1/8	1/16	
RNN	0.4684	0.4139	0.3618	0.3234	0.2830	
LSTM	0.4674	0.4102	0.3547	0.3248	0.3023	
GRU	0.4582	0.4037	0.3471	0.3175	0.2960	
Flashback++(RNN)	0.4982	0.4506	0.4172	0.3942	0.3858	
Flashback++(LSTM)	0.4981	0.4514	0.4165	0.3936	0.3864	
Flashback++(GRU)	0.4931	0.4497	0.4087	0.3926	0.3853	

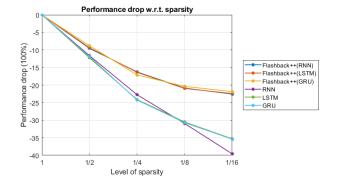


Fig. 10. Performance drop at different levels of sparsity from the original (unsampled) trajectories

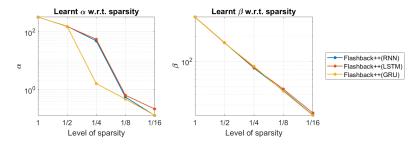


Fig. 11. Learnt  $\alpha$  and  $\beta$  at different levels of sparsity

111:20 Deng et al.

Method	Train-testing splitting ratios						
Method	0.5/0.5	0.6/0.4	0.7/0.3	0.8/0.2	0.9/0.1		
RNN	0.2391	0.2556	0.2652	0.2830	0.2798		
LSTM	0.2812	0.2879	0.2912	0.3023	0.2987		
GRU	0.2666	0.2784	0.2882	0.2960	0.2887		
Flashback++(RNN)	0.3695	0.3730	0.3769	0.3858	0.3782		
Flashback++(LSTM)	0.3733	0.3768	0.3781	0.3864	0.3789		
Flashback++(GRU)	0.3708	0.3740	0.3774	0.3853	0.3770		

Table 8. Performance on UM-WiFi datasets of different train-testing splitting ratios

Table 9. Coefficient of variation in MRR over different train-testing splitting ratios

Method	RNN	LSTM	GRU
Basic RNNs	6.82%	2.89%	4.01%
Flashback++	1.63%	1.27%	1.43%

Table 7 shows the performance of basic RNNs and Flashback++. We observe that all methods yield decreasing performance when the level of sparsity increases, as it is more difficult to model user mobility over more sparse trajectories. Our Flashback++ consistently and significantly outperforms basic RNNs across different levels of sparsity. Moreover, to quantitatively evaluate the robustness against different levels of sparsity, for each method, we compute the relative performance drop from the original (unsampled) trajectories at different levels of sparsity. Figure 10 shows the results. We observe that our Flashback++ achieves much smaller performance drops compared to basic RNNs, reducing the performance drop by 31.05% on average over different levels of sparsity. This implies that Flashback++ is more robust than basic RNNs against different levels of trajectory sparsity.

In addition, we also investigate the learnt values of the spatiotemporal decay parameters  $\alpha$  and  $\beta$  at different levels of sparsity. Figure 11 shows the results. We observe that both  $\alpha$  and  $\beta$  decrease when increasing the level of sparsity of input trajectories. Specifically, for more sparse trajectories, the spatial and temporal distances between successive check-ins are larger; subsequently,  $\alpha$  and  $\beta$  are learnt to have smaller values so as to account more for historical hidden states for next location prediction.

# 5.7 Robustness against different train-testing splitting ratios

In this experiment, we evaluate our method against different train-testing splitting ratios on the UM-WiFi dataset with a sampling rate of 1/16. The results are shown in Table 8. We see that as the training ratio increases, the performance of all methods improves in general. However, we observed that the highest performance was achieved at a train-testing splitting ratio of 0.8/0.2, while the performance drops at a train-testing splitting ratio of 0.9/0.1, due to overfitting. Notably, Flashback++ consistently and significantly outperformed basic RNNs across various train-testing splitting ratios.

Table 9 presents the coefficient of variation across different train-testing splitting ratios for each method. We see that Flashback++ shows significantly lower coefficients of variation compared to basic RNNs (68.37% lower on average). These results indicate that Flashback++ is more robust than basic RNNs against different train-testing splitting ratios.

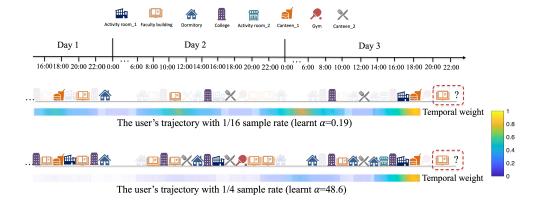


Fig. 12. Case study of Flashback++ on one user trajectory with two different sparsity levels.

## 5.8 Case Study

To future demonstrate how our Flashback++ can learn to optimally integrate the two universal mobility laws for location prediction (i.e., "flash back to the right moment"), we present a case study in Figure 12. We compare the same user's trajectory with two different sparsity levels (under 1/16 and 1/4 sample rates, respectively). In this setting, we investigate the case where the underlying mobility patterns of the two trajectories are the same, but just observed differently.

For each trajectory, we plot the corresponding temporal weights as a color bar using the temporal decay rate  $\alpha$  learnt by our Flashback++. We observe that Flashback++ learns a smaller value of  $\alpha$  for the more sparse trajectory (with 1/16 sample rate); subsequently, the temporal weights for older check-ins become larger, as shown in Figure 12. In other words, for more sparse trajectories, Flashback++ learns to resort more to historical hidden states for prediction. In contrast, for denser trajectories, Flashback++ can effectively model the sequential pattern, and thus learn to use less information from historical hidden states for prediction. In summary, our Flashback++ can automatically adapt these cases with the learnable decay rates.

### 6 CONCLUSION

This paper introduced Flashback++, a general RNN architecture designed for modeling sparse user mobility trajectories, leveraging rich spatiotemporal contexts to robustly overcome the sparsity issue of user mobility trajectories by learning to flash back in hidden states of RNNs. Departing from the widely adopted scheme of adding spatiotemporal factors into the recurrent hidden state passing process of the RNNs, our method explicitly uses the spatiotemporal context to search past hidden states with high predictive power; to fully benefit from the rich spatiotemporal contexts, Flashback++ is guided by the two universal mobility laws, i.e., temporal periodicity and spatial regularity, and optimally integrates them into the RNN architecture by learning to re-weighting the past hidden states for next location prediction, i.e., learning to "flash back to the right moment" under the two mobility laws for location prediction. Our extensive evaluation compares Flashback++ against a sizable collection of state-of-the-art techniques on two real-world LBSN datasets and one on-campus mobility dataset. Results show that Flashback++ not only consistently and significantly outperforms all baseline techniques by 20.56% to 44.36%, but also achieves better robustness of location prediction performance against different model settings (different RNN architectures and numbers of hidden states to flash back), different levels of trajectory sparsity, and different train-testing splitting ratios than baselines, yielding an improvement of 31.05% to 94.60%.

111:22 Deng et al.

In future work, we plan to study the effect of adding social context to further improve the location prediction performance.

#### **ACKNOWLEDGMENTS**

This project has received funding from the University of Macau (MYRG2022-00048-IOTSC), the Science and Technology Development Fund, Macau SAR (0038/2021/AGJ, SKL-IOTSC(UM)-2021-2023), UIC Research Grant (UICR0700021-22), and the European Research Council (ERC, grant agreement 683253/GraphInt). This work was performed in part at SICC which is supported by SKL-IOTSC, University of Macau.

#### REFERENCES

- [1] Nur Al Hasan Haldar, Jianxin Li, Mark Reynolds, Timos Sellis, and Jeffrey Xu Yu. 2019. Location prediction in large-scale social networks: an in-depth benchmarking study. *The VLDB Journal* 28, 5 (2019), 623–648.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).
- [3] Yu Cao, Ang Li, Jinglei Lou, Mingkai Chen, Xuguang Zhang, and Bin Kang. 2021. An Attention-Based Bidirectional Gated Recurrent Unit Network for Location Prediction. In 2021 13th International Conference on Wireless Communications and Signal Processing (WCSP). IEEE, 1–5.
- [4] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI*.
- [5] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In KDD. ACM, 1082–1090.
- [6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078 (2014).
- [7] Qiang Cui, Yuyuan Tang, Shu Wu, and Liang Wang. 2019. Distance2Pre: Personalized Spatial Preference for Next Point-of-Interest Prediction. In *PAKDD*. Springer, 289–301.
- [8] Christos Doulkeridis, Akrivi Vlachou, Nikos Pelekis, and Yannis Theodoridis. 2021. A Survey on Big Data Processing Frameworks for Mobility Analytics. *ACM SIGMOD Record* 50, 2 (2021), 18–29.
- [9] Brian S Everitt and Anders Skrondal. 2010. The Cambridge dictionary of statistics. (2010).
- [10] Jie Feng, Yong Li, Zeyu Yang, Qiang Qiu, and Depeng Jin. 2020. Predicting human mobility with semantic motivation via multi-task attentional recurrent networks. IEEE Transactions on Knowledge and Data Engineering (2020).
- [11] 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.
- [12] Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. 2017. Poi2vec: Geographical latent representation for predicting future visitors. In AAAI.
- [13] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized ranking metric embedding for next new POI recommendation. In IJCAI.
- [14] Flavio Figueiredo, Bruno Ribeiro, Jussara M Almeida, and Christos Faloutsos. 2016. TribeFlow: Mining & predicting user trajectories. In WWW. 695–706.
- [15] Huiji Gao, Jiliang Tang, and Huan Liu. 2012. Exploring social-historical ties on location-based social networks. In ICWSM.
- [16] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. Nature 453, 7196 (2008), 779.
- [17] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian Jensen. 2018. Learning to route with sparse trajectory sets. In 2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE, 1073–1084.
- [18] Peng Han, Shuo Shang, Aixin Sun, Peilin Zhao, Kai Zheng, and Panos Kalnis. 2019. AUC-MF: point of interest recommendation with AUC maximization. In 2019 IEEE 35th international conference on data engineering (ICDE). IEEE, 1558–1561.
- [19] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.
- [20] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735-1780.
- [21] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. Ieee, 263–272.

- [22] Dejiang Kong and Fei Wu. 2018. HST-LSTM: A Hierarchical Spatial-Temporal Long-Short Term Memory Network for Location Prediction.. In IJCAI. 2341–2347.
- [23] Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. 2013. Geo topic model: joint modeling of user's activity area and interests for location recommendation. In WSDM. ACM, 375–384.
- [24] Juha K Laurila, Daniel Gatica-Perez, Imad Aad, Olivier Bornet, Trinh-Minh-Tri Do, Olivier Dousse, Julien Eberle, Markus Miettinen, et al. 2012. The mobile data challenge: Big data for mobile computing research. Technical Report.
- [25] Xixi Li, Ruimin Hu, Zheng Wang, and Toshihiko Yamasaki. 2021. Location Predicts You: Location Prediction via Bi-direction Speculation and Dual-level Association. In IJCAI. 529–536.
- [26] Yang Li, Tong Chen, Yadan Luo, Hongzhi Yin, and Zi Huang. 2021. Discovering collaborative signals for next POI recommendation with iterative Seq2Graph augmentation. arXiv preprint arXiv:2106.15814 (2021).
- [27] Defu Lian, Yongji Wu, Yong Ge, Xing Xie, and Enhong Chen. 2020. Geography-aware sequential location recommendation. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 2009–2019.
- [28] Defu Lian, Xing Xie, Vincent W Zheng, Nicholas Jing Yuan, Fuzheng Zhang, and Enhong Chen. 2015. CEPR: A collaborative exploration and periodically returning model for location prediction. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 1 (2015), 1–27.
- [29] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 2014. GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 831–840.
- [30] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In AAAI.
- [31] Xin Liu, Yongjian Yang, Yuanbo Xu, Funing Yang, Qiuyang Huang, and Hong Wang. 2022. Real-time POI recommendation via modeling long-and short-term user preferences. *Neurocomputing* 467 (2022), 454–464.
- [32] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment* 10, 10 (2017), 1010–1021.
- [33] Zhidan Liu, Zengyang Gong, Jiangzhou Li, and Kaishun Wu. 2020. Mobility-aware dynamic taxi ridesharing. In 2020 IEEE 36th International Conference on Data Engineering (ICDE). IEEE, 961–972.
- [34] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. 2021. Stan: Spatio-temporal attention network for next location recommendation. In Proceedings of the Web Conference 2021. 2177–2185.
- [35] Shuo Ma, Yu Zheng, and Ouri Wolfson. 2013. T-share: A large-scale dynamic taxi ridesharing service. In 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, 410–421.
- [36] Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. In *UbiComp.* ACM, 911–918.
- [37] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In INTERSPEECH.
- [38] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. 2016. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In NIPS. 3882–3890.
- [39] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. 2012. Mining user mobility features for next place prediction in location-based services. In *ICDM*. IEEE, 1038–1043.
- [40] Tieyun Qian, Bei Liu, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2019. Spatiotemporal representation learning for translation-based poi recommendation. TOIS 37, 2 (2019), 18.
- [41] Xuan Rao, Lisi Chen, Yong Liu, Shuo Shang, Bin Yao, and Peng Han. 2022. Graph-flashback network for next location recommendation. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 1463–1471.
- [42] Ernest George Ravenstein. 1885. The laws of migration. Journal of the statistical society of London 48, 2 (1885), 167–235.
- [43] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. AUAI Press, 452–461.
- [44] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In WWW. ACM, 811–820.
- [45] Adam Sadilek, Henry Kautz, and Jeffrey P Bigham. 2012. Finding your friends and following them to where you are. In WSDM. ACM, 723–732.
- [46] Lei Shi, Congcong Huang, Meijun Liu, Jia Yan, Tao Jiang, Zhihao Tan, Yifan Hu, Wei Chen, and Xiatian Zhang. 2020. UrbanMotion: Visual analysis of metropolitan-scale sparse trajectories. *IEEE Transactions on Visualization and Computer Graphics* 27, 10 (2020), 3881–3899.
- [47] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, and Ke Xu. 2022. Unified Route Planning for Shared Mobility: An Insertion-based Framework. ACM Transactions on Database Systems (TODS) 47, 1 (2022), 1–48.

111:24 Deng et al.

[48] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1633.

- [49] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. 2011. Human mobility, social ties, and link prediction. In *KDD*. Acm, 1100–1108.
- [50] En Wang, Yiheng Jiang, Yuanbo Xu, Liang Wang, and Yongjian Yang. 2022. Spatial-Temporal Interval Aware Sequential POI Recommendation. In 2022 IEEE 38th International Conference on Data Engineering (ICDE). IEEE, 2086–2098.
- [51] Gang Wang, Sarita Y Schoenebeck, Haitao Zheng, and Ben Y Zhao. 2016. "Will Check-in for Badges": Understanding Bias and Misbehavior on Location-Based Social Networks. In Tenth International AAAI Conference on Web and Social Media
- [52] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. 2015. Regularity and conformity: Location prediction using heterogeneous mobility data. In *KDD*. ACM, 1275–1284.
- [53] Zhaobo Wang, Yanmin Zhu, Qiaomei Zhang, Haobing Liu, Chunyang Wang, and Tong Liu. 2022. Graph-enhanced spatial-temporal network for next POI recommendation. ACM Transactions on Knowledge Discovery from Data (TKDD) 16, 6 (2022), 1–21.
- [54] Junhang Wu, Ruimin Hu, Dengshi Li, Lingfei Ren, Wenyi Hu, and Yilin Xiao. 2022. Where have you been: Dual spatiotemporal-aware user mobility modeling for missing check-in POI identification. *Information Processing & Management* 59, 5 (2022), 103030.
- [55] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based poi embedding for location-based recommendation. In *CIKM*. ACM, 15–24.
- [56] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudré-Mauroux. 2020. Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States!. In IJCAI. 2184–2190.
- [57] Dingqi Yang, Bin Li, and Philippe Cudré-Mauroux. 2016. POIsketch: Semantic Place Labeling over User Activity Streams. In IJCAI. 2697–2703.
- [58] Dingqi Yang, Bingqing Qu, and Philippe Cudre-Mauroux. 2020. Location-Centric Social Media Analytics: Challenges and Opportunities for Smart Cities. *IEEE Intelligent Systems* 36, 5 (2020), 3–10.
- [59] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. 2019. Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In WWW. ACM, 2147–2157.
- [60] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudré-Mauroux. 2020. Lbsn2vec++: Heterogeneous hypergraph embedding for location-based social networks. IEEE Transactions on Knowledge and Data Engineering (2020).
- [61] Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang. 2013. A sentiment-enhanced personalized location recommendation system. In HT.
- [62] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. 2015. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. TSMC 45, 1 (2015), 129–142.
- [63] Song Yang, Jiamou Liu, and Kaiqi Zhao. 2022. GETNext: trajectory flow map enhanced transformer for next POI recommendation. In Proceedings of the 45th International ACM SIGIR Conference on research and development in information retrieval. 1144–1153.
- [64] Jihang Ye, Zhe Zhu, and Hong Cheng. 2013. What's your next move: User activity prediction in location-based social networks. In SDM. SIAM, 171–179.
- [65] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. 2013. LCARS: a location-content-aware recommender system. In KDD. ACM, 221–229.
- [66] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*.
- [67] Pengpeng Zhao, Anjing Luo, Yanchi Liu, Fuzhen Zhuang, Jiajie Xu, Zhixu Li, Victor S Sheng, and Xiaofang Zhou. 2020. Where to go next: A spatio-temporal gated network for next poi recommendation. IEEE Transactions on Knowledge and Data Engineering (2020).
- [68] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. 2019. Where to go next: A spatio-temporal gated network for next POI recommendation. In AAAI, Vol. 33. 5877–5884.
- [69] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. 2008. Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*. 312–321.
- [70] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM.. In *IJCAI*. 3602–3608.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009