Mobility Inference on Long-Tailed Sparse Trajectory

LEI SHI, YUANKAI LUO, and SHUAI MA, ACT&BDBC, Beihang University
HANGHANG TONG, Department of Computer Science, University of Illinois at Urbana-Champaign
ZHETAO LI, National & Local Joint Engineering Research Center of Network Security Detection and Protection Technology, Jinan University
XIATIAN ZHANG, Beijing Tendcloud Tianxia Technology Co. Ltd.
ZHIGUANG SHAN, State Information Center

Analyzing the urban trajectory in cities has become an important topic in data mining. How can we model the human mobility consisting of stay and travel states from the raw trajectory data? How can we infer these mobility states from a single user's trajectory information? How can we further generalize the mobility inference to the real-world trajectory data that span multiple users and are sparsely sampled over time?

In this article, based on formal and rigid definitions of the stay/travel mobility, we propose a single trajectory inference algorithm that utilizes a generic long-tailed sparsity pattern in the large-scale trajectory data. The algorithm guarantees a 100% precision in the stay/travel inference with a provable lower bound in the recall metric. Furthermore, we design a transformer-like deep learning architecture on the problem of mobility inference from multiple sparse trajectories. Several adaptations from the standard transformer network structure are introduced, including the singleton design to avoid the negative effect of sparse labels in the decoder side, the customized space-time embedding on features of location records, and the mask apparatus at the output side for loss function correction. Evaluations on three trajectory datasets of 40 million urban users validate the performance guarantees of the proposed inference algorithm and demonstrate the superiority of our deep learning model, in comparison to sequence learning methods in the literature. On extremely sparse trajectories, the deep learning model improves from the single trajectory inference algorithm with more than two times of overall and F1 accuracy. The model also generalizes to large-scale trajectory data from different sources with good scalability.

CCS Concepts: • **Information systems → Data mining**; • **Computing methodologies → Machine learning algorithms;**

Additional Key Words and Phrases: Urban data, trajectory inference, transformer

This work was supported by National Key R&D Program of China (2021YFB3500700), NSFC Grant 62172026, the Fundamental Research Funds for the Central Universities, and SKLSDE.

Authors' addresses: L. Shi, Y. Luo, and S. Ma, ACT&BDBC, Beihang University, Beijing, China; emails: leishi@buaa.edu.cn, yuankai0220@163.com, mashuai@buaa.edu.cn; H. Tong, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana; email: htong@illinois.edu; Z. Li, National & Local Joint Engineering Research Center of Network Security Detection and Protection Technology, Jinan University, Guangzhou, China; email: liztchina@hotmail.com; X. Zhang, Beijing Tendcloud Tianxia Technology Co. Ltd., Beijing, China; email: xiatian.zhang@tendcloud.com; Z. Shan (corresponding author), State Information Center, Beijing, China; email: shanzhiguang@263.net.

2157-6904/2023/01-ART18 $15.00
https://doi.org/10.1145/3563457

ACM Reference format:
Lei Shi, Yuankai Luo, Shuai Ma, Hanghang Tong, Zhetao Li, Xiatian Zhang, and Zhiguang Shan. 2023. Mobility Inference on Long-Tailed Sparse Trajectory. *ACM Trans. Intell. Syst. Technol.* 14, 1, Article 18 (January 2023), 26 pages.
https://doi.org/10.1145/3563457

1 INTRODUCTION

The recent surge of metropolitan-scale human trajectory data, e.g., mobile traces [34], taxi logs [50], Wi-Fi probes, and geo-referenced check-ins [2], paves the way for a fundamental understanding of the human mobility in cities. In both theoretical and empirical studies, the urban trajectory of human is considered as interleaving segments of `stay` and `travel` [5, 6, 15]. The inference of these segments from the raw trajectory data plays a pivotal role in solving many urban analytics tasks. For example, in traffic planning and optimization, the detected travels are used as the training data for the travel time estimation [42, 47]. In trade area analysis, the discovery of user's visits to business sites relies on the segmentation of stays and travels from the trajectory data [31].

In the literature, there is a consensus that the stay segments (also known as the stops) can be defined as the part of the trajectory within a spatially constrained region for a sufficiently long time [6, 17, 30]. Algorithms have been proposed that first partition the trajectory at all the record intervals larger than a spatial threshold and infer the resulting sub-trajectories as stay by the definition [6, 30]. On the other hand, no definition of the travel segment has been formulated on the trajectory data. Existing works mostly assume a dense sampling rate in the trajectory data, i.e., seconds or a few minutes on average between the consecutive trajectory records [19, 33]. For such data, the real-time speed of the trajectory can be calculated, which is used to accurately detect all the travels.

Nevertheless, the metropolitan-scale measurement of human trajectories is often extremely sparse over time for pragmatic constraints such as the power consumption and the user privacy. For the mobile sensing data used in this article, the average record interval is as long as 2 hours, two magnitudes larger than that of most previously considered trajectory datasets in the literature. Existing mobility inference algorithms designed for dense trajectories do not work any more. For instance, two consecutive records in a trajectory with a 24-hour time interval and adjacent spatial locations will be identified as in the same stay segment (Figure 1(c)). In fact, these records could be either two separate stays at home or pass-bys during the daily commute. We are *agnostic* about their mobility given the single trajectory information only.

The inference of stay and travel on sparsely sampled trajectories are highly challenging. First, the real-world human movement is a complex process with varied speeds and spatiotemporal patterns. How can we formulate a comprehensive definition of stays and travels on human trajectory data coping with various applications? Second, with the mobility definition, how do we infer which part of the trajectory as stay or travel using the single trajectory information? How can we design the inference algorithm to work with the metropolitan-scale trajectory data with billions of records? Third, it has been known that human movements exhibit strong regularity (e.g., a 93% predictability [38]). How can we leverage such regularity to overcome the limit of the single trajectory inference?

To answer the aforementioned questions, we make the following contributions in this work:

— **The formal definition** of both stay and travel on the continuous trajectory model using a pair of spatial and temporal parameters. The linkage of this continuous mobility model to the sparse trajectory data is rigorously studied, which helps to formulate our research problem. (Section 2)

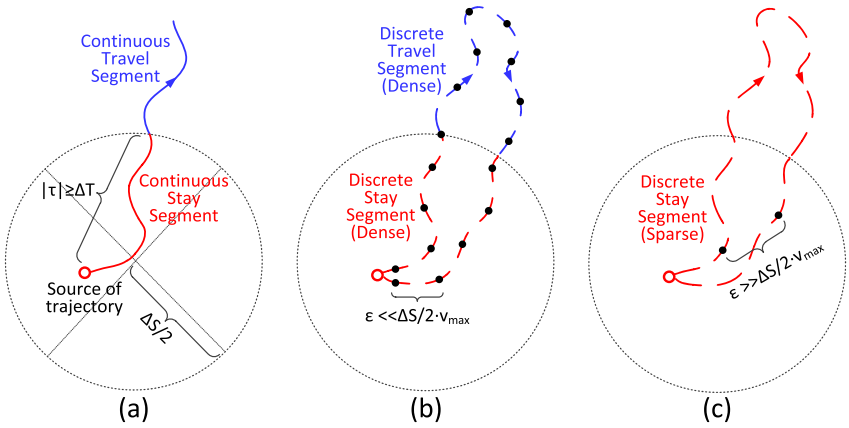

Fig. 1. Illustrative examples of Definitions 1 and 2: (a) continuous stay/travel segments; discrete stay/travel segments on (b) dense trajectory; and (c) sparse trajectory.

— **The single trajectory inference algorithm** called **Slice & Doubly Sliding** (**SDS**) designed according to a generic long-tailed sparsity pattern in our trajectory data (Section 3). The algorithm is proved to guarantee a 100% inference precision and a lower-bounded recall subject to the single trajectory information. (Section 4)

— **The application of an adapted deep learning method** that captures the regularity of human mobility at the population scale. Several neural network features are introduced to cope with the requirement of multiple sparse trajectory inference. These customizations include the singleton transformer design, the specialized space-time embedding of location records, and the mask apparatus at the output side of the network. (Section 5)

We evaluate the proposed SDS algorithm and the singleton transformer deep learning model on both simulated trajectory data and sparse trajectory datasets of 40 million residents in three major Chinese cities (Section 6). The experiment results validate the theoretical performance of the SDS algorithm and demonstrate three key advantages of the deep learning model on mobility inference: the capability to utilize the spatiotemporal information from multiple trajectories, the scalability to large training data, and the generalizability to different urban data sources.

2 PROBLEM DEFINITION

The detailed descriptions of the notations used in this article can be found in Table 1 (for a particular trajectory). We first consider the urban trajectory defined by a continuous mobility model. During a time period T, a user trajectory $\mathbf{\Gamma}$ is composed of a list of temporally continuous records by $\mathbf{\Gamma} = \bigcup_{t\in T} < t, \ell(t) >$. $\ell(t)$ denotes the location of a user at time t.

Definition 1 (Continuous Mobility of a Trajectory). On the continuous segment of the trajectory $\mathbf{\Gamma}$ during a time period $\tau \subseteq T$, denoted as $\boldsymbol{\gamma} = \bigcup_{t\in\tau} < t, \ell(t) >$, we define its mobility by:

(a) $\mathbf{\Gamma}$ is a **stay** segment if: $|\tau| \geq \Delta T$ and $||\ell(t_i) - \ell(t_j)|| < \Delta S$ $(\forall t_i, t_j \in \tau)$;

(b) $\mathbf{\Gamma}$ is a **travel** segment if: $\mathbf{\Gamma}$ does not overlap with any continuous segment satisfying (a).

Here, $|\cdot|$ denotes the length of a time period, $||\cdot||$ is the L_2 norm that computes the spatial distance between two records. ΔT and ΔS are the temporal and spatial parameters in the mobility definition.

As shown by the red curve in Figure 1(a) where the hollow node stands for a long-time stop, Definition 1(a) models the stay segment as a sufficiently long time period ($\geq \Delta T$) when the

Table 1. Notations used in this Article for a Particular Trajectory

Notations	Descriptions
$\|\cdot\|, \|\|\cdot\|\|$	length of a 1D time period, length of a 2D spatial vector
$\Gamma = \bigcup_{t\in T} < t, \ell(t) >$	continuous definition of the trajectory composed of a list of temporally continuous location records
$\gamma = \bigcup_{t\in\tau} < t, \ell(t) > \subseteq \Gamma$	any segment within the continuous trajectory
$T, \tau \subseteq T$	entire time period of the trajectory and any continuous time segment within the period
$\ell(t)$	2D spatial location of the trajectory at a time point t
$\Delta T, \Delta S$	temporal and spatial parameters in the mobility definition
$\Gamma = \bigcup_{t\in\Omega} < t, \ell(t) >$	discrete definition of the trajectory composed of a list of temporally discrete location records
$\gamma = \bigcup_{t\in\omega} < t, \ell(t) > \subseteq \Gamma$	any segment within the discrete trajectory
$\Omega = \{t_1, \ldots, t_L\}$	ordered set of sample time points of the discrete trajectory
$\omega = \{t_p, \ldots, t_q\} \subseteq \Omega$	ordered set of sample time points of any segment of the discrete trajectory
L, ϵ, v_{max}	the number of sample time points, the maximal consecutive time interval, and the maximal movement speed of the discrete trajectory
$I_{S/T}(t)$	continuous mobility state of the trajectory at the time point t (stay or travel)
$\xi(\Gamma), \rho(\Gamma)$	global sparsity and local coverage of the discrete trajectory

trajectory is kept within a circular region of radius $\Delta S/2$. This definition is consistent among all the previous literature [6, 17, 30]. Note that the stay segments by definition can overlap with each other in space and time. Their enclosure is called the maximal stay segment. On the other hand, based on the ground truth that a user can either stay or travel in any time point, the segment not overlapped with any stays is defined as the travel segment (Definition 1(b)).

The continuous mobility model can not be exactly computed in the real world as the human trajectory is hardly measured continuously. In most cases, the trajectory is composed of a list of discrete records on certain time points (e.g., $\Omega = \{t_1, \ldots, t_L\}$) denoted by $\Gamma = \bigcup_{t\in\Omega} < t, \ell(t) >$ where L denotes the sample size of the trajectory. A discrete mobility model can be defined in analogy to the continuous model.

Definition 2 (Discrete Mobility of a Trajectory). On the discrete segment of the trajectory Γ in a time series $\omega = \{t_p, \ldots, t_q\} \subseteq \Omega$ $(1 \le p < q \le L)$, denoted as $\gamma = \bigcup_{t\in\omega} < t, \ell(t) >$, define its mobility by:

(a) γ is stay if: $t_q - t_p \ge \Delta T$ and $||\ell(t_i) - \ell(t_j)|| < \Delta S$ $(\forall t_i, t_j \in \omega)$;

(b) γ is travel if: γ does not overlap with any discrete segment satisfying (a).

The discrete mobility model can be optimally computed by an exact algorithm (Algorithm 2 in Appendix A). Nevertheless, the resulting mobility is not always equivalent to that of the continuous model with the full trajectory information. For example, in Figure 1(b), the stay and travel segments detected on a densely sampled trajectory by the discrete mobility model generally echo those by the continuous model (Figure 1(a)). In comparison, the detected segments shown by Figure 1(c) on the same but sparsely sampled trajectory turn out to be erroneous and largely deviate from the continuous model. The theorem below reveals the relationship between the two models.

Theorem 1 (Intrinsic Linkage Between Discrete and Continuous Mobility of a Trajectory). *Consider a discrete segment γ of the trajectory. Let ϵ be the maximal time interval between the consecutive records of γ, v_{max} be the maximal movement speed in γ:*

(a) γ satisfying Definition 2(a) under the parameters of ΔS and ΔT is also a stay segment by Definition 1(a) in the continuous model under the parameters of $\Delta' S = \Delta S + 2 \cdot \epsilon \cdot v_{max}$ and $\Delta' T = \Delta T$;

(b) γ satisfying Definition 2(b) under the parameters of ΔS and ΔT is also a travel segment by Definition 1(b) in the continuous model under the parameters of $\Delta' S = \Delta S$ and $\Delta' T = \Delta T + 2 \cdot \epsilon$.

Table 2. The Metadata of Each Urban Trajectory Record

Field	Description	Sample
Time	Timestamp of record	18:02:41/07/12/2016
Lon.	Longitude of location	116.523625
Lat.	Latitude of location	39.792935
Mid	Unique device ID	1370021020431

The proof is given in Appendix A. By Theorem 1, for the discrete trajectory satisfying $\epsilon << \min(\frac{\Delta S}{2 \cdot v_{max}}, \frac{\Delta T}{2})$, i.e., having a dense sampling rate, the discrete mobility of the trajectory computed by the exact algorithm can approximate its continuous mobility with tiny parameter changes. Unfortunately, the measurement of human trajectories in big cities is often extremely sparse over time for pragmatic constraints such as the power consumption and the user privacy (e.g., the datasets in Section 3.1). This work studies the inference of the continuous mobility from the sparse trajectory, which can not be approximated by Theorem 1.

Problem (Mobility Inference on Sparse Trajectory).

Given: (1) *a set of urban users*; (2) *each user's sparse trajectory* $\Gamma = \bigcup_{t \in \Omega = \{t_1, \ldots, t_L\}} < t, \ell(t) >$ *that* $\exists j \in [1, L), ||t_j - t_{j+1}|| > \min(\frac{\Delta S}{2 \cdot v_{max}}, \frac{\Delta T}{2})$; (3) *the parameters of* ΔS *and* ΔT *that define the mobility of the trajectory.*

Infer: *the continuous mobility of the sparse trajectory at the time of each record, which is denoted by* $I_{S/T}(t_i), \forall i \in [1, L]$.

Note that the parameters of ΔS and ΔT determine the spatiotemporal scale of mobility. Unless otherwise noted, we use $\Delta S = 800\ m$, $\Delta T = 30\ min$ to study the intra-city mobility. The parameter selection is discussed in Appendix B.

3 SPARSITY ANALYSIS ON TRAJECTORIES

By Theorem 1, our research problem seems intractable on sparse trajectories. In this section, we analyze a set of real-world trajectory data and discover a generic sparsity pattern that can be utilized in accurately inferring the human mobility.

3.1 Data Source

The trajectory data are provided by a mobile analytics company that keeps track of billions of smart devices in China, including mobile phones, tablets, wearable devices, and so on. The company's third-party APIs are registered inside more than 100,000 types of mobile apps in a wide spectrum of domains. When a registered app is activated on a device (not necessarily being used), the API will report the location of that device to the company server. The metadata of each trajectory record is shown in Table 2.

We extract the full-scale trajectory data within three major Chinese cities during a period of 90 consecutive days in 2016, as shown in Table 3. The datasets are immensely huge, e.g., in Beijing it captures the trajectory of 31.8 million devices, which accounts for ~50% of the city's population. The spatial precision of each record is kept within 100 meters, by using the records collected by GPS and Wi-Fi.

3.2 The Long-Tailed Sparsity Pattern

We study the sampling statistics of the trajectory data. The time intervals between consecutive records inside the trajectory are averaged to ~2.5 hours in all the three datasets. With these extremely sparse trajectories, it seems impossible to infer their continuous mobility. As a reference,

Table 3. Three Trajectory Datasets Used in This Work

City	#Device	#Record	Size	Length
Beijing	31,849,742	8,407,648,917	738.1 G	90 days
Tianjin	8,011,128	2,858,575,880	206.8 G	90 days
Tangshan	2,786,668	920,364,499	64.8 G	90 days

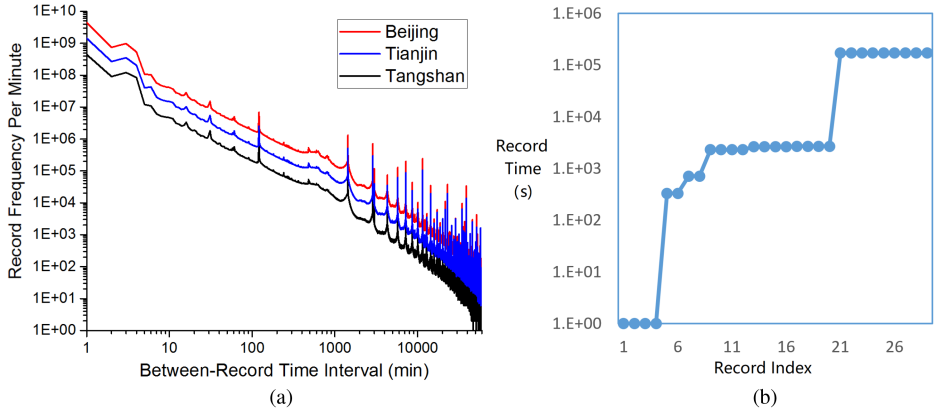

Fig. 2. The long-tailed sparsity pattern: (a) the distribution of between-record time intervals; (b) an example trajectory.

most journeys in a city elapse no longer than 2 hours, during which less than one record is reported on average.

Taking a closer look, we identify that the sampling pattern in our dataset, though sparse, is highly skewed. Figure 2(a) depicts the distributions of between-record time intervals in all trajectories, which follow power-law like decays in the log-log scale. We call this pattern the long-tailed sparsity: Most intervals are very short, while there are also quite a few extremely long intervals that contribute to the large average. The long-tailed sparse trajectory is then defined as the trajectory that fits this pattern. Take the dataset in Beijing as an example, 88.9% intervals are smaller than 30 minutes (a typical ΔT for Definition 1). At the trajectory level, it is observed that most trajectories are composed of multiple densely sampled segments that are far apart from each other over time. An example trajectory is depicted in Figure 2(b).

To capture the long-tailed sparsity pattern, we define two metrics on each trajectory. These metrics are shown later to correlate with the capability for the mobility inference.

Definition 3 (Sparsity Metrics of a Trajectory). For any trajectory Γ observed at $\Omega = \{t_1, \ldots, t_L\}$:

(a) **global sparsity** is the average time interval between the consecutive records of Γ: $\xi(\Gamma) = E(t_{i+1} - t_i), \forall i \in [1, L-1]$;

(b) **local coverage** is the ratio of the records within the dense segment: $\rho(\Gamma) = \frac{L - I(t_i - t_{i-1} > \Delta T, t_{i+1} - t_i > \Delta T)}{L}, \forall i \in [2, L-1]$.

Here, $E(\cdot)$ and $I(\cdot)$ denote the mean function and the size of a set. Note that the global sparsity is independent of the temporal parameter ΔT, while the local coverage is related to ΔT.

We compute the sparsity metrics in the trajectory data of Beijing under five parameter settings ($\Delta T = 5, 10, 15, 30, 45$ minutes). The distribution of the metrics are depicted in Figure 3. The global sparsity in Figure 3(a) follows a power-law like decay similar to the distribution of the between-record intervals in Figure 2(a). The distribution of the local coverage in Figure 3(b) shows

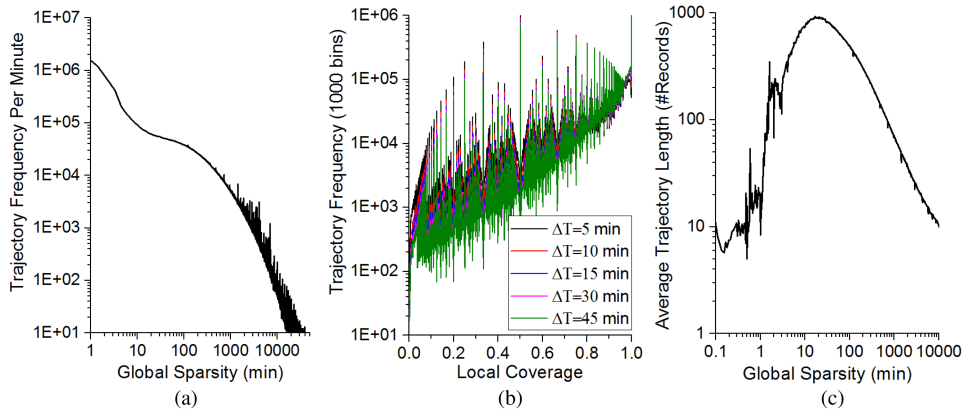

Fig. 3. The distribution of the sparsity metrics in the data of Beijing: (a) global sparsity; (b) local coverage; and (c) average trajectory length by global sparsity.

an exponentially increasing pattern that most of the trajectories have a high local coverage (with an average of 0.897 at $\Delta T = 30$ minutes). This demonstrates that most of the records in the long-tailed sparse trajectory are in the densely sampled segment of the trajectory. As shown in Figure 3(c), the trajectories with extremely low and high sparsity tend to be smaller in length, i.e., the densely sampled short snippets or a few long-distance samples of a trajectory. The trajectories in both cases are therefore exempted from the subsequent analysis.

Note that the long-tailed sparsity pattern is also found in other datasets and application domains. For example, Gonzalez et al. studied the mobile phone user's trajectory data where the location of the user is reported upon each phone call or text message [15]. The time intervals between consecutive records follow a long-tailed power-law decay. In a recent work, Chen et al. analyzed the sparsely sampled geo-tagged social media data [7]. The distributions of the time interval and distance between records follow power-law decays within the space and time scale of a single trip (1 day, 500 km).

4 SINGLE TRAJECTORY INFERENCE

We propose the mobility inference algorithm on the single trajectory. The main idea is to leverage the long-tailed sparsity pattern discovered in our trajectory data (Section 3.2). Though the average record interval in a trajectory is too large to apply Theorem 1, each trajectory can be decomposed into multiple densely sampled segments, on which the continuous mobility can be confidently inferred.

Definition 4 (Dense Stay Segment). A discrete segment γ of the trajectory Γ defined in the time series $\omega = \{t_p, \ldots, t_q\}$ $(1 \le p < q \le L)$ is a **dense stay segment** of Γ if:

(a) γ is a stay segment of Γ by Definition 2(a);

(b) any consecutive time interval of γ is small enough: $\forall p \le i < q,\ t_{i+1} - t_i \le \Delta T$. ΔT is the parameter used in Definition 2(a).

Observation 1 (Continuous Stay Assumption). *Consider a dense stay segment γ detected from the long-tailed sparse trajectory, which is defined in the time series $\omega = \{t_p, \ldots, t_q\}$ $(1 \le p < q \le L)$. For any unobserved time point $t \in (t_i, t_{i+1}), \forall p \le i < q$, we hypothesize that $||\ell(t) - \ell(t_i)|| < \Delta S$ and $||\ell(t) - \ell(t_{i+1})|| < \Delta S$.*

Observation 1 states that if a user is observed frequently in a region of diameter ΔS, any intermediate location between observations is also within a similarly constrained region. We empirically validate this observation by the experiment in Appendix B on our trajectory datasets. The probability of violating the observation is below 10^{-5} in most cases. When Observation 1 holds, we can develop two theorems that characterize the continuous mobility of stay and travel on long-tailed sparse trajectories.

Theorem 2 (Continuous Mobility of Dense Stay Segments). *In the long-tailed sparse trajectory Γ:*

(a) any dense stay segment γ satisfying Definition 4 under the parameters of $\Delta S/3$ and ΔT is also the stay segment by Definition 1(a) in the continuous model under the parameters of ΔS and ΔT;

(b) the continuous mobility of any discrete segment γ in the time period $\tau \in [t_p, t_q]$ can be inferred as stay by Definition 1(a) under the parameters of ΔS and ΔT only if γ defined in $\omega = \{t_p, \ldots, t_q\}$ is the dense stay segment under the same parameters.

Theorem 3 (Continuous Mobility of Travel Records). *Consider a discrete trajectory Γ defined in the time series $\Omega = \{t_1, \ldots, t_L\}$:*

(a) any record at time t_i $(1 < i < L)$ is in the travel segment by the continuous model of Definition 1(b) under the parameters of ΔS and ΔT if only there exist $1 \le p < i < q \le L$ that: (1) $||\ell(t_i) - \ell(t_p)|| \ge \Delta S$; (2) $||\ell(t_i) - \ell(t_q)|| \ge \Delta S$; (3) $t_q - t_p \le \Delta T$;

(b) any record at time t_i can be inferred as in the travel segment by Definition 1(b) under the parameters of ΔS and ΔT only if there exist $1 \le p < i < q \le L$ that: (1) $||\ell(t_i) - \ell(t_p)|| \ge \Delta S/2$; (2) $||\ell(t_i) - \ell(t_q)|| \ge \Delta S/2$; (3) $t_q - t_p \le \Delta T$.

The proofs are given in Appendix A. By Theorems 2 and 3, we design a new algorithm to infer the continuous mobility of a single long-tailed sparse trajectory, called SDS. As shown in Algorithm 1, the algorithm first slices the trajectory into multiple dense segments at all the intervals larger than ΔT (L2). On each dense segment γ, the stay/travel segments are detected respectively (L3∼10, L11∼19). In particular, the stay detection checks all the segments of γ with the condition in Definition 4 under the parameters of $\Delta S/3$ and ΔT by Theorem 2. To avoid the worst-case $O(L^4)$ complexity, we introduce a doubly sliding window data structure which keeps track of the currently checked segment. The key of the algorithm lies in that, when one pair of records no closer than $\Delta S/3$ are found (L6), all the segments containing this pair of records will be pruned early in the detection and the sliding window will advance aggressively (L10). The travel detection follows Theorem 3. The average-case complexity of SDS is $O(L \cdot W)$ where W is the average number of records in a maximal stay segment.

According to Theorems 2(a) and 3(a), the SDS algorithm guarantees a 100% precision in the mobility inference of both stay and travel. By Theorems 2(b) and 3(b), the lower bounds of the recalls in detecting the stay and travel are $\frac{SDS(\Gamma,S,\Delta S/3,\Delta T)}{SDS(\Gamma,S,\Delta S,\Delta T)}$ and $\frac{SDS(\Gamma,T,\Delta S,\Delta T)}{SDS(\Gamma,T,\Delta S/2,\Delta T)}$, respectively, where $SDS(\Gamma, S/T, \Delta S, \Delta T)$ are the number of stay and travel records detected by the SDS algorithm from Γ under the parameters of ΔS and ΔT. Note that the recall is defined on all the stay and travel records that can be detected given the single sparse trajectory, not on the continuous mobility of records given the full trajectory information. Another advantage of the SDS algorithm lies in that it also works for dense trajectories, each of which is treated as one densely sampled segment.

We applied the SDS algorithm to our dataset in Beijing. 47.0%∼50.2% and 0.044%∼0.83% records are detected as stay and travel, depending on the parameters of ΔS and ΔT. Figure 4 shows the average stay/travel percentages by the global sparsity of a trajectory. All the curves are bell-shaped with only one peak: The highest ratio of stay is found at the global sparsity around 1.6 min (97.3%∼98.5%, Figure 4(a)); the highest ratio of travel is found at the global sparsity from 5 to

ALGORITHM 1: SDS on long-tailed sparse trajectories.

Input : $\Gamma = \bigcup_{i\in[1,L]} < t_i, \ell(t_i) >, t_1 < \cdots < t_L$ (sparse trajectory), ΔT, ΔS (space, time parameters)
Output : $I_{S/T}(t_i), \forall i \in [1, L]$ (mobility of each record)

begin
  /* Γ into M segments (γ_j) at every interval larger than ΔT */
  $\{\gamma_j = \{< t_{j,k}, \ell(t_{j,k}) >\}_{k\in[1,L_j]}\}_{j=[1,M]} \leftarrow Divide(\Gamma, \Delta T)$
  for $j \leftarrow [1, M]$, $head \leftarrow 1$ **do**
    /* detect all the stay segments on γ_j */
    for $cursor \leftarrow [2, L_j]$ **do**
      /* iterate all the records backward from $cursor - 1$ */
      for $anchor \leftarrow [cursor - 1, head]$ **do**
        /* cut at the first escape outside the range of $\frac{\Delta S}{3}$ */
        if $||\ell(t_{j,cursor}) - \ell(t_{j,anchor})|| \geq \frac{\Delta S}{3}$ **then**
          /* stay segment */
          if $t_{j,cursor-1} - t_{j,head} \geq \Delta T$ **then**
            for $k \leftarrow [head, cursor - 1]$ **do**
              $I_{S/T}(t_{j,k}) \leftarrow$ S
          $head \leftarrow anchor + 1$, **Break**
    /* detect all the travel records on γ_j */
    for $cursor \leftarrow [2, L_j - 1]$ && $I_{S/T}(t_{j,cursor}) \neq S$ **do**
      /* find the first left record outside the range of ΔS */
      for $l \leftarrow [cursor - 1, 1]$ **do**
        if $||\ell(t_{j,cursor}) - \ell(t_{j,l})|| \geq \Delta S$ **then**
          $left \leftarrow l$, **Break**
      /* find the first right record outside the range of ΔS */
      for $r \leftarrow [cursor + 1, L_j]$ **do**
        if $||\ell(t_{j,cursor}) - \ell(t_{j,r})|| \geq \Delta S$ **then**
          $right \leftarrow r$, **Break**
      if $t_{j,right} - t_{j,left} \leq \Delta T$ **then**
        $I_{S/T}(t_{j,cursor}) \leftarrow$ T
  return $I_{S/T}(t_i), i = [1, L]$

10 min, which increases with ΔT (0.34%~3.8%, Figure 4(b)). Before the peak of stay, the trajectory is mostly composed of less than 10 records (Figure 3(c)), with a time period shorter than ΔT and can not be inferred as stay. After the peaks of stay and travel, the ratio of detected records drops due to the increased sparsity of the trajectories. This validates Theorem 1 that sparser trajectories are harder for the continuous mobility inference.

By the empirical result, the parameters of $\Delta T = 30min$ and $\Delta S = 800m$ are chosen and used throughout this work. The details are explained in Appendix B. We also note that though SDS is designed to infer the continuous mobility of a single trajectory, the same SDS algorithm can be applied to infer the discrete mobility of the same trajectory, using different parameters: $(\Delta S, \Delta T)$ for the stay state of discrete mobility, $(\Delta S/2, \Delta T)$ for the travel state of discrete mobility, according to Theorems 2 and 3. In addition, although we use the classical machine learning metrics of precision and recall to evaluate the SDS algorithm, the algorithm itself is not a machine learning-based method. It is deterministic and the 100% precision is achieved by definition.

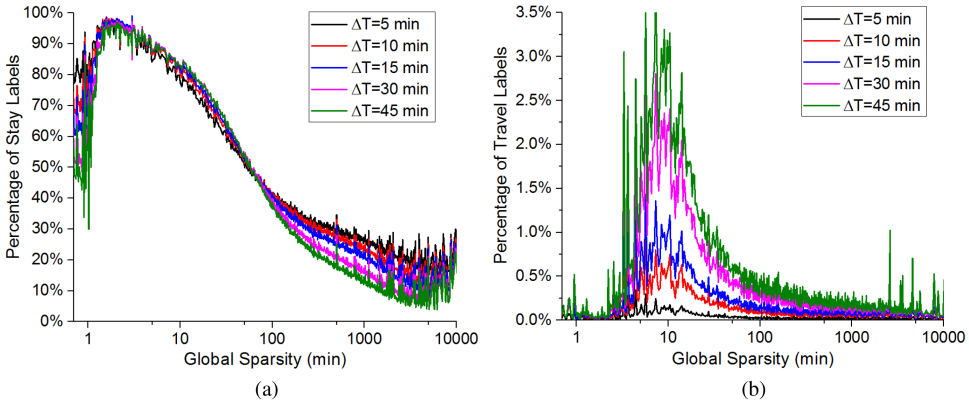

Fig. 4. Percentages of stay/travel on the trajectory with different global sparsity values and ΔT (ΔS fixed to 800 m).

5 MULTIPLE TRAJECTORY INFERENCE

The single trajectory inference algorithm (i.e., SDS) can correctly predict the continuous mobility of about a half records in each trajectory. For the other half of records, it is infeasible to infer the mobility from the single trajectory information only because the conditions in Theorems 2(b) and 3(b) do not hold for these records. In addition to using data from the single trajectory, multiple trajectory inference also considers the similarity of different trajectories. To take multiple trajectories as input for mobility inference, we propose to employ deep learning methods, which have been proven to deliver good performance given large amount of training data. It is expected that the multi-trajectory model could learn the spatiotemporal regularity of human mobility, beyond the theoretical mobility definition used in the SDS algorithm.

As mobility state inference is a typical sequence-to-sequence learning problem, we consider the deep learning methods developed for processing sequential data. In the literature, most state-of-the-art deep neural network architectures for sequence modeling have been applied to the problem of next-location prediction (see Section 7.3 for details), which is quite close to our problem. Nevertheless, the next-location prediction only accepts input sequences precedent to the currently predicted location, while our problem allows to incorporate the full trajectory information. To leverage this advantage, we decide to apply the encoder-decoder architecture that computes full representation on each trajectory. In addition to this difference, there are also several special challenges in designing the deep learning architecture for our problem. For example, we do not have 100% labels in the output side of the training data because the SDS algorithm only labels discrete location records on each trajectory. This brings difficulty both in the decoder side of the architecture and in specifying the cost function of the entire model. In the following, we first propose a transformer-based deep learning architecture and then introduce several customized designs to tackle the special challenges of our problem.

5.1 Singleton Transformer Architecture

As shown in Figure 5(a), we adopt a transformer-like sequence-to-sequence deep learning architecture. The raw trajectory is labeled by the SDS algorithm for the true stay/travel state before input to the neural network. Notably, we introduce a mechanism to truncate variable-length long trajectories into fixed-length sub-trajectories. This is because typical sequence-to-sequence deep learning models generally take a sequence of tokens up to several hundreds (e.g., a long sentence of 100 words). While in our problem, the user trajectory could be much longer, up to 5,000 records.

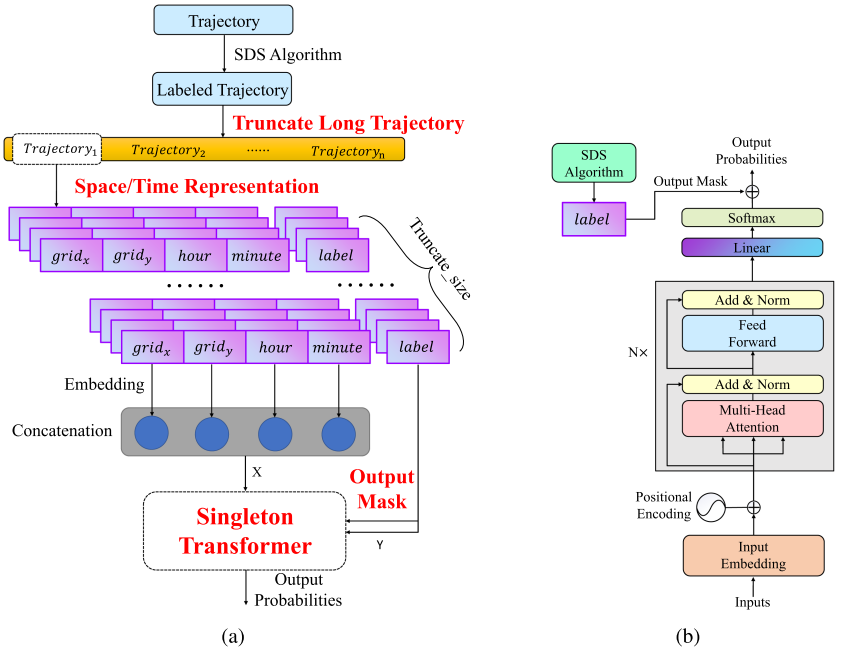

Fig. 5. The sequence-to-sequence deep learning architecture for multiple trajectory inference: (a) the overall neural network structure with several design customizations (in red text); (b) the singleton transformer with self-attention mechanism.

With sequence truncating, local context can be trained on each sub-trajectory while still preserving the expressibility for inference, as proposed by Luong et al. [27]. After that, the location records in each sub-trajectory are embedded by exploiting customized space-time representations for our task (Section 5.2). The latest transformer architecture is applied in the final stage to construct the multiple trajectory inference model.

In the deep learning community, the transformer [44] is preferred over classical encoder-decoder architectures using RNNs/LSTMs/GRUs [40] because the transformer design better captures distant dependencies within a long trajectory. Also, its self-attention neural network can be trained and applied more efficiently via parallel processing as the sequential, recurrent operations in conventional encoder-decoders are avoided. In our mobility inference setting, we find that the standard transformer design does not work quite well. It is mainly because the mobility state labels in the training data are largely discrete, leaving about a half of records unlabeled. Providing these sparse labels to the decoder side of transformer disrupts the consistency of training context and leads to poor performance (see Section 6.2 for results).

We propose a **singleton transformer** (**STF**) design (Figure 5(b)) that removes the decoder network and only leaves the encoder network taking the trajectory representation as input. The key mechanisms in transformer remain largely unchanged in our design. The core self-attention module in the center of Figure 5(b) adopts the scaled dot-product attention function to alleviate the effect of large dot-product. The multi-head attention mechanism is also employed to exploit the full representation capability on input trajectory. The self-attention is further enhanced by residual connection, layerwise normalization, and a stacked design to amplify the nonlinearity of representation. Finally, we introduce a masking apparatus in the output side of the STF architecture to cope with the unlabeled training data, which is detailed in Section 5.3.

5.2 Space-Time Embedding

In our trajectory data, both space and time information are recorded in high resolutions, i.e., by a millionth of degree (longitude/latitude) and a millisecond (timestamp). Adding these raw inputs to the neural network could affect the model performance in converging and learning. After examining the mobility inference task, we extract several task-relevant spatial and temporal features from the raw input, discretize their values, and compute vector embeddings to represent their essential information. The embeddings are updated online during the training process.

In more detail, spatially we divide the territory of each city into grids of 0.001 degree latitude/longitude. The location of each record is converted to the latitude and longitude indices of the grid it belongs to. Each grid index is represented by a vector of length E. Temporally, we divide the timestamp of each record into two indices and embed them separately. The first is the absolute hour index of the timestamp. For example, the timestamp of 12:50 a.m. on Monday has an index of $12 + 24 \cdot 1 = 36$. The hour index indicates the time of day and the day of the week upon the observation, which can be related to the mobility of the record. The second is the relative minute index defined as the elapsed time in minutes from the start of the current segment in the trajectory. Here, the segments of a trajectory is computed by the SDS algorithm (L2 of Algorithm 1). Both hour and minute indices have a length of E in the embedding. Finally, the space and time embeddings are concatenated into a vector of length $4E$, which is provided as input to the STF architecture. The embedding for each grid and time index is randomized upon the initialization. We use $E = 100$ by default.

5.3 Output Mask

In the output side, we only have the mobility state label (stay or travel) for a subset of records on each trajectory. If we only include these labeled records in the training, the test performance might downgrade because of the loss of consistent context, as empirically shown by the performance of variant of STFs (Section 6.2). In an improved design, we propose to feed the full trajectory to the input side of the neural network, but mask out losses for the records without labels in the output side. This allows the STF network consistently capture the dynamics of the entire trajectory and be trained with supervisions if available.

The final loss function for training is

$$\textbf{Loss} = -\sum_{t \in S_{label}} \mathbf{y}^{(t)} \log p(\hat{\mathbf{y}}^{(t)}), \tag{1}$$

where $\mathbf{y}^{(t)}$ denotes the mobility label at the time step t, $\hat{\mathbf{y}}^{(t)}$ denotes the prediction result at the time step t, and S_{label} is the set of record indices labeled by the SDS algorithm.

6 EVALUATION

6.1 Experiment Setup

Data. We evaluate the SDS algorithm and the proposed deep learning model on three types of data extracted from the raw data in Section 3.1.

— ***Full*** data are a set of randomly selected trajectories. We apply the SDS algorithm to create the stay/travel labels on each trajectory. Because the travel labels are rare (<1%) and a large percentage of short trajectories have no travel label at all, we only select the trajectories with at least 10 travel labels. Note that this criterion does not lead to a biased selection for the mobility inference. The eligible trajectories have an average global sparsity mildly smaller than the average in all the datasets. We extract 2×3 groups of non-overlapping 10 K, 40 K, and 100 K trajectories for train and test, respectively. They are called `FU-10K`, `FU-40K`, and

FU-100K. By default, the dataset collected at Beijing is used. Only labeled records on the full data are evaluated.
— ***Re-sampled*** data are used to evaluate the inference performance on unlabeled records. Given a trajectory from the full data, we randomly keep each record with a probability (i.e., the re-sampling rate). The re-sampled training data is re-labeled by the SDS algorithm, normally generating a smaller percentage of labels than the full data. On the re-sampled test data, we re-use the labels generated in the full data. The series of data re-sampled from FU-10K is called RE-10K. All the records with labels in the full data can be evaluated.
— ***Simulated*** data are used to evaluate the SDS algorithm itself, as no true label can be detected beyond this algorithm. The simulated data is generated using the timestamps in the full data and then re-sampled. More details of the data generation are described in Appendix C. The labels of all the records in the simulated data are known and can be evaluated.

Method. Nine alterative methods except for the proposed STF model are applied to the mobility inference problem. Four are deep learning methods with different architecture:

— ***Transformer.*** The original transformer model by Vaswani et al. [44]. The trajectories and their mobility labels are used as input in the encoder and decoder side of the neural network. At the decoder side, a mask apparatus is also applied to bypass unlabeled records in self-attention.
— ***DWSTTN.*** The Deep Wide Spatio Temporal Transformer Network [1] is a transformer-like encoder-decoder architecture designed for the prediction of a taxi's next destination. In both encoder and decoder sides, taxi's trajectory information is embedded and injected, which is different from the original transformer using only labels in the decoder side. Notably, each transformer module is split into two neural networks to embed spatial and temporal information separately.
— ***ST-LSTM .*** The **Spatial-Temporal Long-Short Term Memory** (**ST-LSTM**) [18] enhanced over the original LSTM network for the sequence learning problem of location records. It introduces spatial-temporal factors into the basic gate mechanisms of LSTM. Spatial-temporal relations in the trajectory data are embedded to mitigate the sparsity effect. Note that ST-LSTM leverages **Area of Interest** (**AOI**) information, which is unavailable and replaced by the spatial-temporal grid index in our setting.
— ***LSTM.*** The classical sequence deep learning method using unidirectional LSTM cells. Only labeled records are fed as input into both train and test.

Another two methods are customized classifiers for spatial-temporal trajectory data proposed in the literature:

— ***CB-SMoT.*** A speed-based spatial-temporal clustering approach [29], which could also be used for mobility inference over trajectory data (See Section 7.2 for details). The algorithm parameters are set to $\Delta S = 800\ m$, $\Delta T = 30\ min$.
— ***ST-DBSCAN.*** A density-based spatial-temporal clustering approach for trajectory data [4] (See Section 7.2 for details). The algorithm parameters are set to $\Delta S = 800\ m$, $\Delta T = 30\ min$, $MinPts = 3$, according to the heuristics given in Ester et al.'s work [10].

Finally, the other three generic machine learning classifiers are compared. To apply these classifiers, we conduct window-based feature extraction. On each location record of a trajectory, W records before and after the record are selected, which should also locate within the dense segment containing the record (generated by L2 of Algorithm 1). Then, a feature vector of length $(2W+1)\cdot 4$ is formed in which four spatial-temporal indices are used to represent each record (see Section 5.2

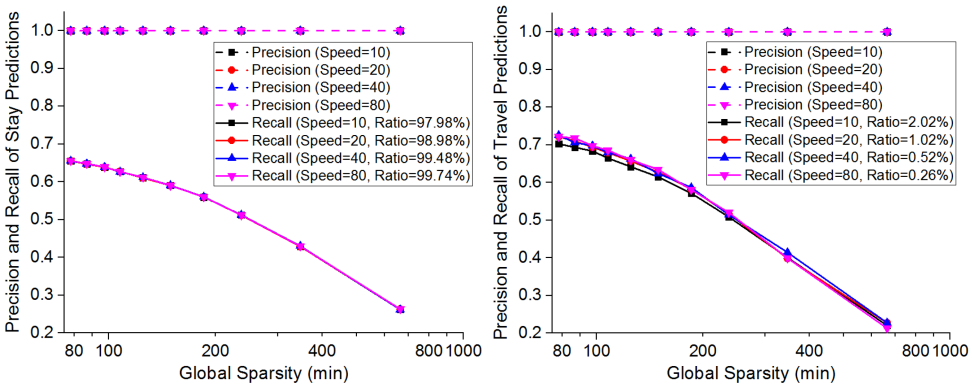

Fig. 6. SDS inference on simulated data: (a) stay; (b) travel.

for details). We test through $W = 1 \sim 10$ and find that $W = 4$ achieves the best tradeoff between performance and cost for most classifiers.

- ***LG.*** It applies the logistic regression over the extracted features.
- ***DT.*** It applies the decision tree over the extracted features.
- ***HMM.*** The stay/travel label is used as the state of Hidden Markov Models. The spatio temporal offset between consecutive records is used as the observation. The prediction is computed by the Viterbi algorithm [45]. The method mimics the technique in the next location prediction using Markov chains [13].

In each experiment, we measure the *P*recision and *R*ecall in predicting the *S*tay and tra*V*el labels separately, which are denoted as SP, SR, VP, VR. The overall prediction accuracy of the two classes of labels is denoted as ACC. As the distribution of label classes is unbalanced (much fewer travels than stays), we also report $F1 - ACC$, the harmonic mean of the F1 measures in stay and travel predictions. The source code is available at https://github.com/LUOyk1999/MobilityInference.

6.2 Quantitative Result

SDS algorithm. We evaluate the SDS algorithm on the simulated data over `RE-10K`, with re-sampling rates ranging from 1.0 to 0.1. As shown in Figure 6, the precision of both stay and travel predictions (dashed lines) is 100%, regardless of the re-sampling rate and the speed used in the simulation. This validates the theoretical result in Section 4. As the re-sampling rate decreases, which leads to a linear increase in the global sparsity by Definition 3(a) (X-axis), the recall drops at a rate slightly slower than the empirical result in Figure 4.

Deep learning architecture. We evaluate five design choices of the singleton transformer on the `FU-10K` dataset: # of attention heads (4), # of stacked attention layers (3), the dropout probability (0.1), the truncate size on long trajectories (200), and the use of mask in the output side (with). The experiment results are listed in Table 4 and the best hyperparameters are given in the parentheses above. It shows that multi-layer multi-head self-attention with a moderate dropout works better in our scenario. The use of output mask and truncating to smaller segments also improve the prediction performance.

Model comparison. On the labeled records of `FU-10K` data, Table 5 compares the performance of all the multiple trajectory inference methods. The proposed STF model performs the best in overall metrics of ACC and $F1-ACC$ (0.97 and 0.95). The standard **transformer model** (**TF**) has a poor result in the recall of travel records (0.26) because of the lost of consistent labeled context in the

Table 4. Performance with Different Transformer Hyperparameters on FU-10K Data

		# Heads			# Layers				Dropout Prob.				Truncate Size			Mask	
		1	2	4	1	2	3	4	0.1	0.2	0.3	0.4	100	200	300	with	w/o
Pre.	Stay(SP)	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97	0.98	0.99	0.98	0.93
	Travel(VP)	0.89	0.92	0.91	0.86	0.90	0.91	0.91	0.91	0.91	0.93	0.93	0.91	0.91	0.89	0.91	0.99
Rec.	Stay(SR)	0.98	0.99	0.98	0.97	0.98	0.98	0.98	0.98	0.98	0.99	0.99	0.98	0.98	0.98	0.98	0.998
	Travel(VR)	0.88	0.89	0.91	0.90	0.91	0.91	0.91	0.91	0.91	0.87	0.88	0.85	0.91	0.93	0.91	0.59
	ACC	0.96	0.97	**0.972**	0.96	0.97	**0.972**	0.971	**0.972**	0.971	0.97	0.97	0.96	**0.972**	0.97	**0.972**	0.93
	$F1$	0.93	0.94	**0.95**	0.92	0.94	**0.95**	0.94	**0.95**	0.94	0.94	0.94	0.93	**0.95**	0.95	**0.95**	0.84

Best result in bold text.

Table 5. Comparison of Inference Methods on FU-10K Data

	STF	TF [44]	DWSTTN [1]	ST-LSTM [18]	LSTM	LG	DT	HMM	CB-SMoT	ST-DBSCAN
SP	0.98	0.87	0.94	0.92	0.93	0.84	0.93	0.85	**0.99**	0.92
VP	0.91	0.77	0.95	**0.97**	0.80	0.29	0.50	0.86	0.45	0.25
SR	0.98	0.99	0.99	0.99	0.97	**1.00**	0.87	**1.00**	0.88	0.91
VR	**0.91**	0.26	0.68	0.53	0.59	0.00	0.66	0.10	**0.91**	0.29
ACC	**0.97**	0.87	0.94	0.93	0.91	0.84	0.84	0.85	0.88	0.85
$F1$	**0.95**	0.55	0.87	0.80	0.79	0.00	0.69	0.30	0.73	0.42

Best result in bold text.

decoder side. The similar transformer-based DWSTTN model gets the second highest performance, still 10 % worse than STF in $F1-ACC$. Among other classifiers, ST-LSTM, LSTM, CB-SMoT, and DT perform moderately and achieve an $F1-ACC$ of 0.69~0.8, respectively. Notably, CB-SMoT obtains the best SP and VR by a biased classification toward travel states, which is penalized by a VP of only 0.45. The other models (ST-DBSCAN, LG, HMM) all perform badly in predicting the travels, with a VP or VR smaller than 0.3. Though LG gets the best SR, it achieves that by classifying all the records as stay.

Extending to the unlabeled part of the RE-10K data, we summarize the performance comparison with datasets under decreasing re-sampling rates in Figure 7. These re-sampled data also correspond to trajectory input with increasing global sparsity (refer to Figure 6). Because the stay records are generally well-predicted by most methods, we only depict ACC, VP, VR, and $F1-ACC$. Note that the performance of the SDS algorithm is also plotted, serving as the upper bound achieved with the single trajectory information only. The STF model is still the best in most metrics when the re-sampling rate is higher than 0.1, except that DWSTTN, ST-LSTM, and HMM has high precisions on much smaller portion of travel records ($VR < 0.5$ for most cases). CB-SMoT and ST-DBSCAN obtain high VRs while having low VP.

The proposed STF model surpasses the SDS algorithm on ACC from the re-sampling rate of 0.8, starting to enjoy the bonus of multiple trajectory information. Even on the metric of $F1-ACC$ favoring the travel prediction, STF outperforms SDS from the re-sampling rate of about 0.6. Although SDS guarantees a 100% VP according to the theoretical result in Section 4, its travel prediction performance dives quickly with re-sampling rates below 0.5. Also, the trajectory completion technique [20] does not improve the inference performance. Even worse, because the technique needs to pre-compute a junction network using spatially dense trajectories as input, the test data before completion is constrained into a 5×5 km^2 square region. The precision and recall of SDS in spatially constrained datasets is worse than the randomly sampled FU-10K data.

We try to improve the mobility inference by using the densely sampled trajectory as the training data, i.e., the 100% re-sampled data in RE-10K; and test on the sparse trajectory, i.e., RE-10K with re-sampling rates of 0.1~1. This is realistic in the model building. As shown in Figure 8, with denser trajectories and more labels in the training, the test performance of the STF model (straight lines in cyan with symbols) is improved from the model with sparse input (cyan lines without symbols),

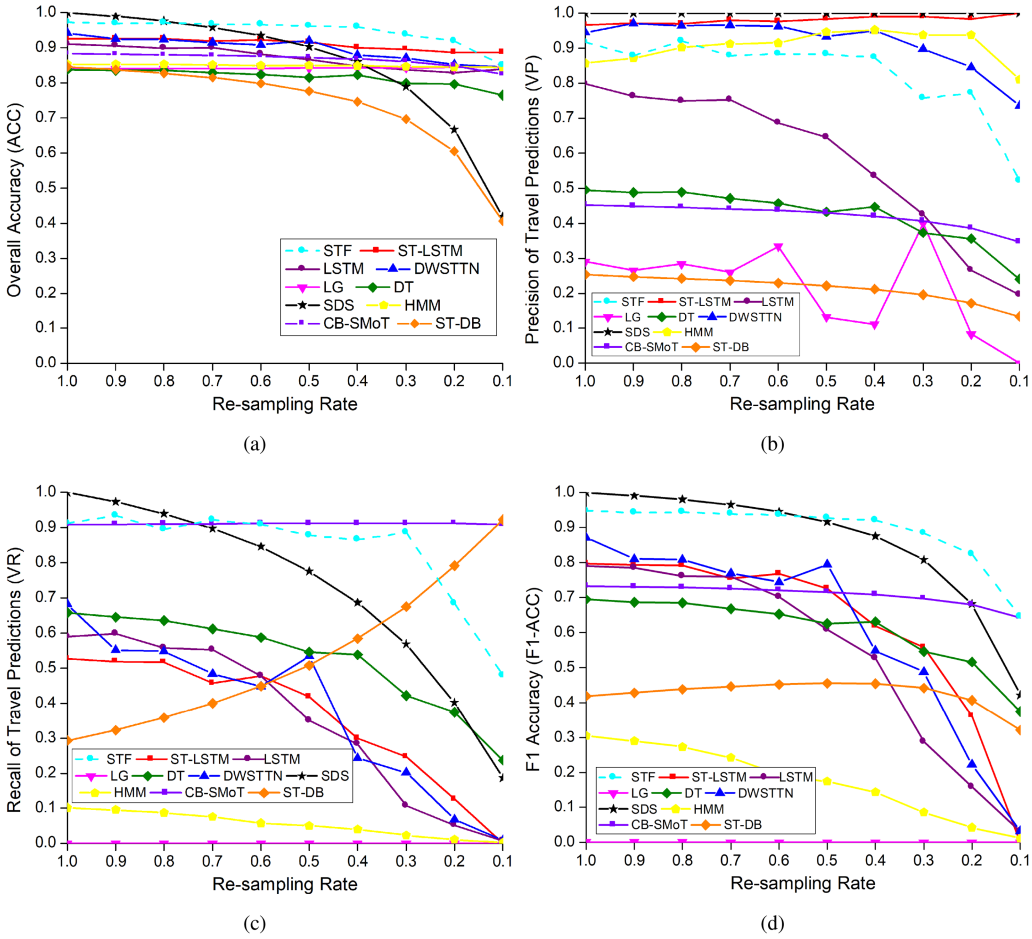

Fig. 7. Comparison of inference methods on RE-10K data. Dashed lines are the proposed STF model.

especially in the travel prediction and the re-sampling rates below 0.4. Taking this finding one step further, we use 100K 100% re-sampled trajectories in training (the RE-100K data). The result is much better—the STF model outperforms the upper bound of SDS from the re-sampling rate of 0.9. In a 10% re-sampling setting, the STF model achieves 2.24 × *ACC* and 2.08 × $F1 - ACC$ compared with SDS.

Scalability and generalizability. We carry out the same experiment on the full dataset with higher numbers of trajectories. As shown in Table 6, the performance keeps steady using the same FU-10K as the training data and test on 10 K, 40 K, and 100 K full trajectory data (FU-10K:10K, etc.). This shows that the model trained on a small dataset can be generalized to much larger datasets. Training on the larger data further improves the test performance, which is nearly optimal for FU-100K:10K ($ACC = 0.995$, $F1 - ACC = 0.99$).

We conduct the same experiment on RE-10K with the dense trajectory input, using the datasets in Tianjin and Tangshan. Compared with Figure 8(b) for Beijing, the $F1 - ACC$ of the STF model in Tianjin shows a similar curve (Figure 9(a)), surpassing the SDS from a re-sampling rate of 0.6. On the other hand, the STF model does not work better than the SDS on the Tangshan data (Figure 9(b)). We hypothesize that this is because the selected train/test data in Tangshan has a much lower

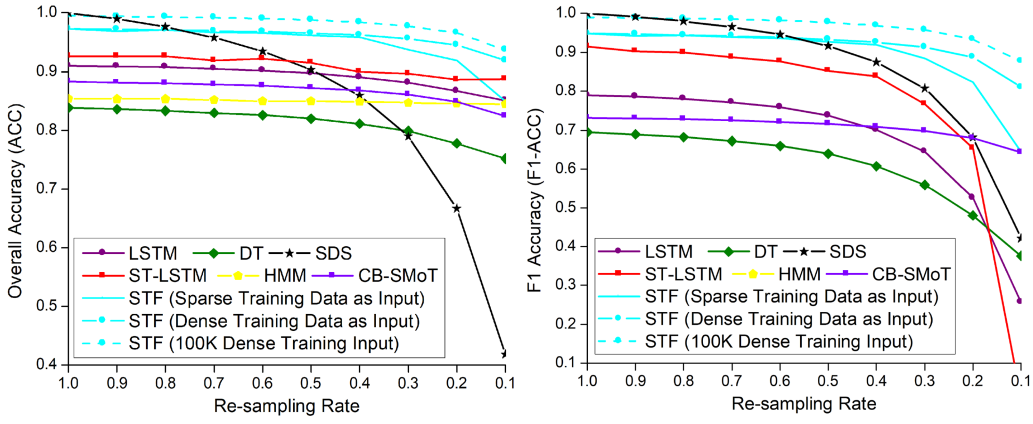

Fig. 8. Training with 100% RE-10K data and test on re-sampled RE-10K data. Cyan lines are STF models.

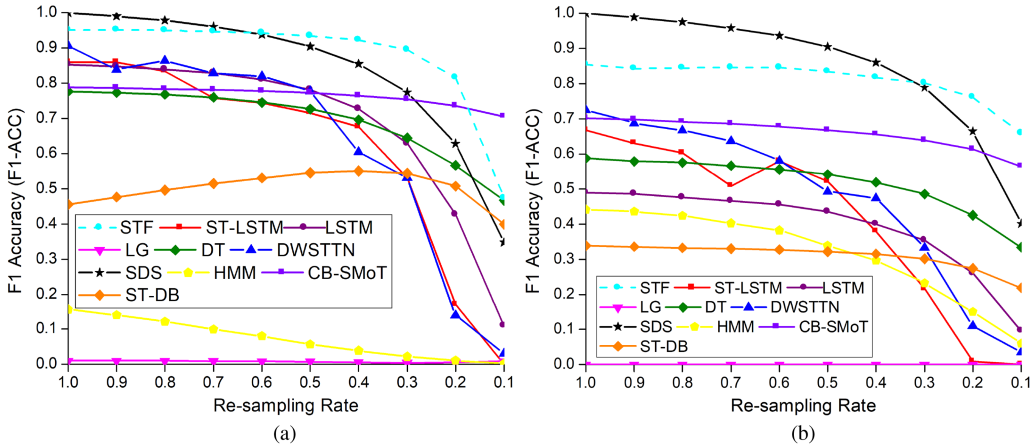

Fig. 9. Experiments with RE-10K: (a) Tianjin; (b) Tangshan.

Table 6. The Scalability of the Proposed Deep Learning Model

	10K	10K:40K	10K:100K	100K:10K	100K:40K	100K:100K
SP	0.98	0.98	0.98	**0.998**	0.997	0.997
VP	0.91	0.91	0.89	**0.98**	0.98	0.98
SR	0.98	0.98	0.98	**0.996**	0.996	0.996
VR	0.91	0.92	0.92	**0.99**	0.99	0.98
ACC	0.97	0.97	0.97	**0.995**	0.99	0.99
$F1$	0.95	0.95	0.94	**0.99**	0.99	0.99
Time (s)	**0.87k**	0.94k	1.15k	8.78k	8.78k	9.86k

Best result in bold text.

percentage of travel labels (5.56%) than Beijing (10.64%) and Tianjin (16.94%). The model can not learn useful patterns given fewer labels. Tangshan is also a smaller city than Beijing and Tianjin, where we have fewer data (Table 3).

Implications. The experiment result demonstrates that the SDS algorithm is accurate on the single trajectory (100% SP and VP). The optimized deep learning models can learn from the multiple trajectory input to improve the single trajectory mobility inference through the excellent

generalizability to sparse trajectories and the scalability to large training data. In fact, we expect the proposed model to perform even better in comparison to the SDS algorithm. We only evaluate on the part of the trajectory labeled in the 100% re-sampled test data. For the unlabeled test data (38.9% for Beijing), it is reasonable to guess that our model performs similarly to the labeled part, while the SDS can not infer at all. In future, we plan to develop the re-sampling mechanism on the simulated data to test on the 100% labels of the trajectory dataset.

7 RELATED WORK

7.1 The Study of Urban Trajectory

Using the trajectory data in the city to understand the urban activity and human mobility has been a recent focus of study [51]. On the continuously measured trajectory, the detailed route information is available for analysis [21, 22, 50]. For instance, the trajectories of taxis can be used to classify drivers by their job performance [22] or aggregated as time-dependent landmark graph [50] and trajectory visualization [21], in order to compute the fastest route for drivers. Based on over one million bank note circulation reports in US, Brockmann et al. explained the human mobility as the combination of a scale-free jump and a heavy-tailed wait, and proposed a random-walk model to characterize these findings [5]. The group of Barabási explained the high degree of spatiotemporal regularity in human mobility by the tendency to avoid visiting new places and to return to the previously visited locations [15, 37].

On the analysis of urban trajectories, the need for separating stay and travel has been partially met by the greedy algorithms similar to Algorithm 2 in Appendix A [6, 17]. Nevertheless, none of these works formally define the stay/travel state of a trajectory, nor do they consider the mobility inference problem on sparse trajectories.

7.2 The Inference of Sparse Trajectory

There are two definitions of the sparse trajectory in the literature of urban data analysis. The first one considers a sequence of infrequent reports from vehicles, e.g., the floating car data recording the location, speed, direction, and time information of the vehicle [43]. These trajectories are usually collected in a uniform time interval of seconds or a few minutes. We call them the temporally sparse trajectory [19, 33, 36]. The second class is the spatially sparse trajectory data in which many road segments in a city are not covered by any of the trajectory, especially for a given period of time. For a queried path of the road network, there is often no such trajectory in the data exactly following the path. The literature on this class of data mostly studied the travel time estimation problem [19, 36, 47].

We mainly consider the mobility inference problem on the temporally sparse trajectory, as our dataset covers most of the city regions. The recent works on this topic focus on the extraction of travel paths [19, 33]. Typically, the problem is decomposed into two tasks: the map-matching and the path-inference. In map-matching, each location record on the trajectory is matched to a point on a particular segment of the road network [32]. In path-inference, the matched points on the map are connected by shortest paths to form the travel path [25].

The map-matching-based techniques can not be applied directly to our mobility inference problem. First, the trajectory data in our case encompasses not only the movement of high-speed vehicles on the ground, but also those by bikes and subways. The locations of these trajectories may not be on the road network, thus are not appropriate for map-matching. It is also costly to evolve the technique with the fast-changing road network of modern cities. Second, we have both travel and stay in our data, while the previous approaches mostly work on the travel part of the trajectory with a temporal sparsity two orders of magnitudes smaller than our case. The trajectory completion techniques can also be applied to compute dense trajectories from known sparse

ones. In Yang et al.'s work [20], a geometry-based method was proposed which pre-computed the junction networks in cities and then predicted the missing part of the travel trajectory, without knowing the city map. However, their technique requires the speed and heading information of each location record, and spatially dense trajectory dataset to pre-compute the junction network. Applying the same trajectory completion technique on our problem leads to worse performance than the proposed deep learning method.

Meanwhile, spatial-temporal clustering algorithms can also be applied for mobility inference of trajectory data. ST-DBSCAN [4] mimicked the classical DBSCAN algorithm. It starts from the first point p of the trajectory and retrieves all points following p in the trajectory that are density-reachable from p. The parameters of ΔT and ΔS in the mobility definition are used as the radius of the neighborhood in DBSCAN (ST-DBSCAN). If p is a core object, a cluster is formed and all points of the cluster are classified as stay records. If p is a border or noise point, then the algorithm iterates to the next point of the trajectory. The algorithm repeats until all points have been processed. Palma et al. proposed another spatial-temporal clustering algorithm called CB-SMoT [29]. The algorithm classifies stay/travel states based on the speed of the trajectory. Similar to ST-DBSCAN, core points are detected, but by the speed of the trajectory with a threshold of $\Delta S/\Delta T$. The points of clusters formed by core points are classified as stay records, while the other points are classified as travel records. We have implemented these algorithms and compared with our approach in the experiments. Results show that clustering algorithms do not perform well in temporally sparse trajectory, as shown in Section 6. At the same time, they are costly in computational complexity, e.g., $O(L \cdot \log L)$ for ST-DBSCAN in average, which is the longest among all methods compared.

7.3 Deep Learning for Human Mobility

Modern deep learning methods have been applied intensively in many analytical tasks related to human mobility [26]. One class of tasks focus on human mobility at the crowd level where the movement pattern is modeled collectively on the group of people, e.g., the crowd flow prediction [16, 39, 52] and the crowd flow generation [24, 49]. In comparison to the crowd-level analysis, the mobility inference problem studied in this article is more relevant to the other class of existing work on individual-level human mobility analysis where every single trajectory is treated separately. The dominant tasks include next-location prediction [3, 9, 14, 28] and trajectory generation [12, 46]. Specially, the next-location prediction problem is defined as the forecast of an individual's next stay location given its historical trajectory data and is the closest to the mobility inference problem studied here. Below we will mainly discuss the literature on the next-location prediction task. For more extended introductions on the field of deep learning for human mobility, we refer to the latest survey by Luca et al. [26].

Some early work applied generic neural network designs to predict the next location on the individual level. De Brébisson et al. [9] presented a multi-layer perceptron architecture with the ReLU activation function that won the taxi destination prediction challenge in the 2015 ECML/PKDD conference. The input data to the architecture include both the trajectory prefix data in GPS points and the metadata of the taxi trajectory such as the departure time, the driver id, and the passenger information. Lv et al. [28] proposed T-CONV, a multi-layer convolutional neural network that models each trajectory as a two-dimensional image. Their design elaborates the multi-scale spatial pattern embedded in the trajectory data to achieve more accurate destination prediction. These traditional deep learning methods, though shown to be effective in particular data and task, do not explicitly take the temporal information of the trajectory into the consideration. In contrast, most recent approaches on next location prediction treat the trajectory as sequence data and apply specialized neural network designs for temporal data, notably RNN, GRU, LSTM, and the encoder-decoder architecture.

Liu et al. [23] introduced ST-RNN that extends the original RNN architecture with time-specific and distance-specific transition matrices to model different time intervals and spatial distances. ST-RNN can therefore embed local temporal and spatial context to better predict individual's next location. Experiment results on benchmark data show that ST-RNN significantly outperforms the original RNN. Feng et al. [11] proposed DeepMove, a hybrid neural network architecture to combine an attention model for periodic patterns from historical trajectories and a GRU model for complicated sequential information from the current trajectory. Kong and Wu [18] developed HST-LSTM with the Spatial Temporal LSTM (ST-LSTM) as its building blocks. ST-LSTM was invented to mitigate the sparsity effect in trajectory data by embedding their spatial-temporal relations. Next, HST-LSTM applies an encoder-decoder architecture over ST-LSTM to finally predict the next location. Rossi et al. [35] proposed a LSTM network equipped with self-attention modules for the next location prediction problem. Importantly, they enhance the semantics of the input trajectory with geographical information from location-based social networks. Yang et al. [48] designed a modified RNN architecture by introducing the flashback mechanism. Flashbacks allow to explicitly search the historical hidden states on the RNN that share similar contexts as the current trajectory. These context and hidden states can then be utilized to better predict the next location of the current trajectory. Chen et al. [8] proposed DeepJMT, a deep model that jointly predict the individual's next location and the arrival time. DeepJMT integrates three neural network modules that include a hierarchical RNN encoder capturing the user's mobility regularities and temporal patterns, spatial/periodicity context extractors modeling patterns on spatial neighbors and periodic trajectories in the history, and a social-temporal context extractor learning from the user's social relationship. Most recently, Tang et al. [41] proposed an integrated deep learning architecture that takes three key mobility data for trip destination prediction of taxi drivers: the partial trajectory of on-going trips, the historical trajectories, and the side information such as driver's characteristics, and land-use information of geo-locations. Both LSTM networks to process sequential data and fully connected neural networks to embed/combine side information are utilized. Abideen et al. [1] introduced the latest transformer design to an encoder-decoder architecture called DWSTTN. Notably the spatial information and temporal information are embedded via two separate transformer modules.

Overseeing the state-of-the-art deep learning literature on next location prediction, the majority of approaches adopt non-encoder-decoder designs that reply on the partial trajectory as input for prediction. A few methods also introduce side information such as social network information and user's demographics which are not available in the large-scale trajectory dataset of our work collected and processed via a privacy-preserving protocol. Importantly, the mobility (state) inference problem studied here is different from the next location prediction in that the full trajectory is feasible and probably viable for learning. This is a missed opportunity for non-encoder-decoder architectures.[1] For these reasons, we conducted experiments to compare our method with two encoder-decoder-based models described above (DWSTTN [1] and ST-LSTM [18]). The experiment results have been reported in Section 6. Notably, the encoder-decoder approach by DeepJMT [8] is not compared as it depends on the user's social relationship which is not ready in our dataset.

8 CONCLUSION

This article studies the problem of mobility inference over sparse trajectories. Based on the observation of a long-tailed sparsity pattern in the trajectory data, we design a single trajectory inference algorithm that detects the mobility of close to half of trajectory records with a guaranteed 100%

[1]Unless a bidirectional LSTM network is adopted, which is not available in the literature because the next location prediction problem does not have access to the full trajectory information.

prediction precision. Furthermore, we apply an adapted singleton transformer deep learning architecture that captures the mobility pattern from multiple trajectories. The proposed deep learning model outperforms baseline models in both deep and conventional machine learning classifiers, as well as customized algorithms for spatiotemporal trajectory data. In particular, by feeding the large-scale densely sampled trajectory data as training input, our model achieves a near-optimal overall accuracy on the records labeled by the single trajectory inference algorithm. On unlabeled records, our model outperforms the single trajectory inference by a factor of two with extremely sparse trajectories as input. Experiment results also demonstrate that the proposed model generalizes to different urban data sources and scales to large datasets.

APPENDICES

A PROOFS AND THE EXACT ALGORITHM

Theorem 1 (Intrinsic Linkage Between Discrete and Continuous Mobility of a Trajectory).

Proof. Theorem 1(a). For the discrete stay segment γ in the time series $\omega = \{t_p, \ldots, t_q\}$ ($t_q - t_p \geq \Delta T$), consider its corresponding continuous segment $\boldsymbol{\gamma'}$ in the time period $\tau = [t_p, t_q]$. $\boldsymbol{\gamma'}$ satisfies $|\tau| = t_q - t_p \geq \Delta T$. $\forall t_i, t_j \in \tau$, we have $||\ell(t_i) - \ell(t_j)|| \leq ||\ell(t_i) - \ell(t'_i)|| + ||\ell(t'_i) - \ell(t'_j)|| + ||\ell(t'_j) - \ell(t_j)||$, given that the straightline is the shortest distance between $\ell(t_i)$ and $\ell(t_j)$. Here, t'_i and t'_j are the closest time point in ω to t_i and t_j respectively. Because $||\ell(t'_i) - \ell(t'_j)|| < \Delta S$, $||\ell(t_i) - \ell(t'_i)|| \leq \epsilon \cdot v_{max}$, $||\ell(t'_j) - \ell(t_j)|| \leq \epsilon \cdot v_{max}$, we have $||\ell(t_i) - \ell(t_j)|| < \Delta S + 2 \cdot \epsilon \cdot v_{max}$. That is, $\boldsymbol{\gamma'}$ is a stay segment by Definition 1(a) under the parameters of $\Delta' S = \Delta S + 2 \cdot \epsilon \cdot v_{max}$ and $\Delta' T = \Delta T$.

Theorem 1(b). For the discrete travel trip γ in the time series $\omega = \{t_p, \ldots, t_q\}$, by definition, we have $\forall t'_i, t'_j \in \omega$ ($t'_j - t'_i \geq \Delta T$), there exist two time points $t'_m, t'_n \in \omega$ ($t'_i \leq t'_m < t'_n \leq t'_j$) satisfying $||\ell(t'_m) - \ell(t'_n)|| \geq \Delta S$. Consider the corresponding continuous segment $\boldsymbol{\gamma'}$ in the time period $\tau = [t_p, t_q]$, $\forall t_i, t_j \in \tau$ ($t_j - t_i \geq \Delta T + 2 \cdot \epsilon$), we can find t'_i (the closest time point in ω no smaller than t_i) and t'_j (the closest time point in ω no larger than t_j), having $t'_j - t'_i \geq \Delta T$. There exist two time points $t'_m, t'_n \in \omega$ ($t_i \leq t'_i \leq t'_m < t'_n \leq t'_j \leq t_j$) satisfying $||\ell(t'_m) - \ell(t'_n)|| \geq \Delta S$. That is, $\boldsymbol{\gamma'}$ is a travel trip by Definition 1(b) under the parameters of $\Delta' S = \Delta S$ and $\Delta' T = \Delta T + 2 \cdot \epsilon$. □

Theorem 2 (Continuous Mobility of Dense Stay Segments).

Proof. Theorem 2(a). For the dense stay segment γ defined in the time series $\omega = \{t_p, \ldots, t_q\}$, consider its corresponding continuous segment $\boldsymbol{\gamma'}$ in the time period $\tau = [t_p, t_q]$. We have $|\tau| = t_q - t_p \geq \Delta T$ because γ is the dense stay segment. For any two time points $t, t' \in [t_p, t_q]$, denote the closest time points in the time series of ω to t and t' as t_i and t_j ($p \leq i \leq q,\ p \leq j \leq q$). We have $||\ell(t) - \ell(t')|| \leq ||\ell(t) - \ell(t_i)|| + ||\ell(t_i) - \ell(t_j)|| + ||\ell(t_j) - \ell(t')|| < \Delta S/3 + \Delta S/3 + \Delta S/3 = \Delta S$ by Observation 1. The conditions for the continuous model of the stay segment in Definition 1(a) then hold.

Theorem 2(b). For the discrete segment γ defined in the time series $\omega = \{t_p, \ldots, t_q\}$, consider its corresponding continuous segment $\boldsymbol{\gamma'}$ in the time period $\tau = [t_p, t_q]$. If $\exists p \leq i < q,\ t_{i+1} - t_i > \Delta T$, i.e., the unobserved time period of (t_i, t_{i+1}) has a duration longer than ΔT. Observing a time period $\tau' \subset (t_i, t_{i+1})$ with $|\tau'| = \Delta T$ can detect a different stay segment from the other part of the segment in (t_i, t_{i+1}). Then, there can be travel trips surrounding the segment in τ' to connect the trajectory. This possibility can not be validated or rejected given the information of the discrete segment γ only. Therefore, the corresponding continuous segment $\boldsymbol{\gamma'}$ can not be inferred as stays, unless $\forall p \leq i < q,\ t_{i+1} - t_i \leq \Delta T$.

On the dense segment γ, if the corresponding continuous segment $\boldsymbol{\gamma'}$ is the stay segment, by Definition 1(a), $\forall t_i, t_j \in \omega \subset \tau$, $||\ell(t_i) - \ell(t_j)|| < \Delta S$. Therefore, γ must be a dense stay segment. □

Theorem 3 (Continuous Mobility of Travel Records).

Proof. Theorem 3(a). For the record at time t_i, consider any time period $\tau = [t, t']$ satisfying $|\tau| \geq \Delta T$ and $t_i \in \tau$. If the three conditions hold, the time period of $\tau' = [t_p, t_q]$ satisfies $|\tau'| \leq \Delta T$ and $t_i \in \tau'$. We have $t_p \in \tau$ or $t_q \in \tau$. Otherwise, we will have $t_p < t$ and $t_q > t'$, which leads to the contradiction of $|\tau'| = t_q - t_p > t' - t = |\tau| \geq \Delta T$. For the either case of $t_p \in \tau$ or $t_q \in \tau$, we have $||\ell(t_i) - \ell(t_p)|| \geq \Delta S$ and $||\ell(t_i) - \ell(t_q)|| \geq \Delta S$. This contradicts to Definition 1(a). Therefore, the record at time t_i can not be in any stay segment, and it must be in a travel trip by Definition 1(b).

Theorem 3(b). For the record at time t_i $(1 < i < L)$, if the condition does not hold, $\forall 1 \leq p < i < q \leq L$ satisfying $t_q - t_p \leq \Delta T$, we have $||\ell(t_i) - \ell(t_p)|| < \Delta S/2$ or $||\ell(t_i) - \ell(t_q)|| < \Delta S/2$.

Consider the smallest time point t_j satisfying $t_j > t_i$ and $||\ell(t_i) - \ell(t_j)|| \geq \Delta S/2$. We should have $t_j - t_i \leq \Delta T$ because otherwise $\forall t_i < t_k < t_j$, $||\ell(t_i) - \ell(t_k)|| < \Delta S/2$. There exists a time period of $\tau' = [t_i, t_j)$, for all observed $t_k \in \tau'$, $||\ell(t_i) - \ell(t_k)|| < \Delta S/2$. Then $\forall t_k, t_{k'} \in \tau'$, $||\ell(t_k) - \ell(t_{k'})|| \leq ||\ell(t_i) - \ell(t_k)|| + ||\ell(t_i) - \ell(t_{k'})|| < \Delta S$, t_i will be possibly in a stay segment, without the information to reject the possibility.

Having $t_j - t_i \leq \Delta T$ and $||\ell(t_i) - \ell(t_j)|| \geq \Delta S/2$, using the proof by contradiction, we have $\forall k < i$ satisfying $t_j - t_k \leq \Delta T$, we have $||\ell(t_i) - \ell(t_k)|| < \Delta S/2$. Consider the largest time point $t_{j'}$ satisfying $t_j - t_{j'} > \Delta T$, we can construct a time period of $(t_{j'}, t_j)$, for all the observed time point of t_k having $j' < k < j$, we have $||\ell(t_i) - \ell(t_k)|| < \Delta S/2$. The distance between these observed time points is below ΔS. Then there can be a continuous segment in the time period of $\tau' \subset (t_{j'}, t_j)$ satisfying $|\tau'| = \Delta T$. We do not have any information to reject the inference of stays on this segment. Therefore, the record at $t_i \in \tau'$ can not be in any travel trip. □

We introduce an exact algorithm to infer the discrete mobility (Definition 2) from densely sampled trajectories, as shown in Algorithm 2. The algorithm iterates all the candidate segments in a trajectory to decide whether they meet the condition of stays. The records not in any stay segments are travels. The algorithm has a computational complexity of $O(L^4)$ (L is the number of records in a trajectory), which is computationally infeasible for the large-scale trajectory data. In our targeted scenario, we do not have the densely sampled trajectory.

B MATERIAL FOR THE SDS ALGORITHM

To validate Observation 1, we conduct an experiment on the full dataset in Beijing. For each record in a trajectory, we explicitly remove the record and detect all the dense stay segments from the remaining trajectory. If the record is within a dense stay segment, we check whether the record, as time t, violates Observation 1. As shown in Figure 10, among 10-billion potential <record, interval> pairs for each parameter setting, the probability of violating Observation 1 is below 10^{-5} if $\Delta S \geq 800m$ and $\Delta T \leq 30min$.

In the mobility definition of the trajectory model, the parameters of ΔS and ΔT need to be determined. In fact, these parameters provide the flexibility to capture the multi-scale mobility in the human trajectory. Inside the city boundary, ΔT and ΔS can be minutes and meters to describe the short-term stays and travels; while in the state level, ΔT and ΔS can be days and hundreds of kilometers to characterize the stay in a city and the travel between cities.

We focus on the detection of intra-city travels because the number of travel records is much fewer than the stay and the detection of stay is relatively insensitive to the parameter change (Figure 4(a), Figure 11(a)). The goal is to detect more travels while keeping the mobility definition reasonable. According to Figure 4(b), we pick $\Delta T = 30min$ because the detected ratio of travel does not increase much when switching to $\Delta T = 45min$ and it does not impose a strict stay definition which violates Observation 1. Similarly, according to Figure 11(b), we pick $\Delta S = 800m$ which

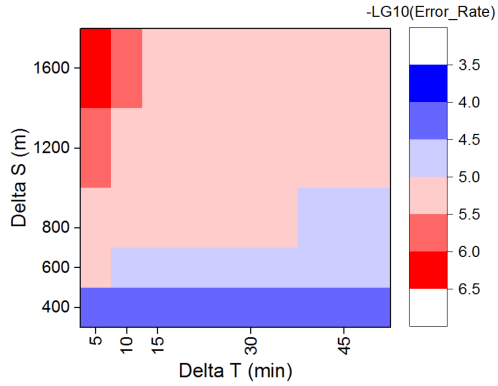

Fig. 10. The probability for violating Observation 1, under different ΔS and ΔT, mapped by the $-\lg_{10}$ operator.

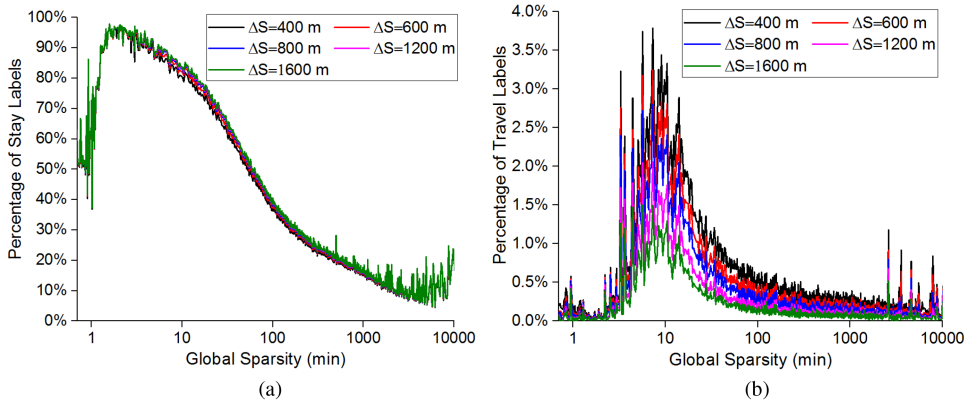

Fig. 11. The percentage of stay/travel on the trajectory with different global sparsity values and ΔS (ΔT fixed to 30 min).

maximizes the recall of travel ($\frac{SDS(\Gamma,T,\Delta S,\Delta T)}{SDS(\Gamma,T,\Delta S/2,\Delta T)}$) and allows a mild stay definition compared with $\Delta S = 400m$. The parameters of $\Delta T = 30min$ and $\Delta S = 800m$ are consistent with the empirical settings in [6, 17].

C GENERATION OF THE SIMULATION DATA

First, we apply the CTRW model in [5, 15] to generate artificial human trajectories. The model characterizes the human trajectory as a two-state interplay between the scale-free displacements (travel) and a long-tailed waiting time distribution (stay). The probability density functions of both the travel distance and the waiting/stay time apply the truncated power-law function observed in [15]. The exponent parameters of the function are calibrated by our trajectory dataset applying the SDS algorithm. Each trajectory starts from a random location. Within each stay period, the location at any time is computed by the stay location plus a random spatial offset smaller than $\Delta S/2$. The travel between consecutive stay locations is assumed to be a straight-line, constant-speed trajectory. The parameter speed controls the ratio of the stay/travel time.

Second, each trajectory is sampled using the timestamps in the full data of Section 6.1. The generated trajectory is further re-sampled by the given re-sampling rate for the real usage in the experiment.

ALGORITHM 2: The exact algorithm on dense trajectories.

Input : $\Gamma = \bigcup_{i \in [1,L]} < t_i, \ell(t_i) >, t_1 < \cdots < t_L$ (dense trajectory), ΔT, ΔS (the space and time parameters)
Output : $I_{S/T}(t_i), \forall i \in [1, L]$ (the mobility of each record)

begin
  for $head \leftarrow [1, L-1]$ **do**
    for $cursor \leftarrow [head + 1, L]$ **do**
      /* iterate all the candidate stay segments */
      if $t_{cursor} - t_{head} \geq \Delta T$ **then**
        for $i \leftarrow [head, cursor - 1]$ **do**
          for $j \leftarrow [i + 1, cursor]$ **do**
            if $||\ell(t_i) - \ell(t_j)|| \geq \Delta S$ **then**
              $Stay \leftarrow$ False, **Break**
        if $Stay! = False$ **then**
          for $i \leftarrow [head, cursor]$ **do**
            $I_{S/T}(t_i) \leftarrow$ S
  /* the remaining records are travel trips */
  for $i \leftarrow [1, L]$ **do**
    if $I_{S/T}(t_i)! = S$ **then**
      $I_{S/T}(t_i) \leftarrow$ T
  return $I_{S/T}(t_i), i = [1, L]$

D REPRODUCIBILITY INFORMATION

Upon the publication of this article, we will provide the software codes and the compiled models including but not limited to the SDS algorithm, the optimized deep learning architecture, the baseline neural network models, and the alternative feature-based classifiers. On the dataset, we will make open a sample sparse trajectory dataset (`FU-10K`) and its labeled version by the SDS algorithm. Because the authors are not permitted to re-post the raw trajectory dataset, the sample trajectories will be open with pre-defined offsets added to both location records (longitude and latitude) and timestamps. It has been tested that these offsets do not affect the model performance used in this work. Upon request, we will point interested researchers to the owner of the raw trajectory data for the permission to access it.

REFERENCES

[1] Zain Ul Abideen, Heli Sun, Zhou Yang, Rana Zeeshan Ahmad, Adnan Iftekhar, and Amir Ali. 2021. Deep wide spatial-temporal based transformer networks modeling for the next destination according to the taxi driver behavior prediction. *Applied Sciences* 11, 1 (2021), 17.

[2] G. Andrienko, N. Andrienko, H. Bosch, T. Ertl, G. Fuchs, P. Jankowski, and D. Thom. 2013. Thematic patterns in georeferenced tweets through space-time visual analytics. *Computing in Science and Engineering* 15, 3 (2013), 72–82.

[3] Yi Bao, Zhou Huang, Linna Li, Yaoli Wang, and Yu Liu. 2021. A BiLSTM-CNN model for predicting users' next locations based on geotagged social media. *International Journal of Geographical Information Science* 35, 4 (2021), 639–660.

[4] Derya Birant and Alp Kut. 2007. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering* 60, 1 (2007), 208–221.

[5] D. Brockmann, L. Hufnagel, and T. Geisel. 2006. The scaling laws of human travel. *Nature* 439, 7075 (2006), 462–465.

[6] Francesco Calabrese, Francisco C. Pereira, Giusy Di Lorenzo, Liang Liu, and Carlo Ratti. 2010. The geography of taste: Analyzing cell-phone mobility and social events. In *Proceedings of the International Conference on Pervasive Computing*. 22–37.

[7] Siming Chen, Xiaoru Yuan, Zhenhuang Wang, Cong Guo, Jie Liang, Zuchao Wang, Xiaolong Luke Zhang, and Jiawan Zhang. 2016. Interactive visual discovering of movement patterns from sparsely sampled geo-tagged social media data. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 270–279.

[8] Yile Chen, Cheng Long, Gao Cong, and Chenliang Li. 2020. Context-aware deep model for joint mobility and time prediction. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 106–114.

[9] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. 2015. Artificial neural networks applied to taxi destination prediction. In *Proceedings of the 2015 International Conference on ECML-PKDD Discovery Challenge* 1526, 40–51.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. 226–231.

[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 *Proceedings of the 2018 World Wide Web Conference*. 1459–1468.

[12] Jie Feng, Zeyu Yang, Fengli Xu, Haisu Yu, Mudan Wang, and Yong Li. 2020. Learning to simulate human mobility. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3426–3433.

[13] 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* 3.

[14] Qiang Gao, Fan Zhou, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2019. Predicting human mobility via variational attention. In *Proceedings of the World Wide Web Conference*. 2750–2756.

[15] Marta C. Gonzalez, César A. Hidalgo, and A.-L. Barabási. 2008. Understanding individual human mobility patterns. *Nature* 453, 7196 (2008), 779–782.

[16] Renhe Jiang, Zekun Cai, Zhaonan Wang, Chuang Yang, Zipei Fan, Quanjun Chen, Kota Tsubouchi, Xuan Song, and Ryosuke Shibasaki. 2021. DeepCrowd: A deep model for large-scale citywide crowd density and flow prediction. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[17] Shan Jiang, Gaston A. Fiore, Yingxiang Yang, Joseph Ferreira Jr, Emilio Frazzoli, and Marta C. González. 2013. A review of urban computing for mobile phone traces: Current methods, challenges and opportunities. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing* 2.

[18] Dejiang Kong and Fei Wu. 2018. HST-LSTM: A hierarchical spatial-temporal long-short term memory network for location prediction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2341–2347.

[19] Weizi Li, Dong Nie, David Wilkie, and Ming C. Lin. 2017. Citywide estimation of traffic dynamics via sparse gps traces. *IEEE Intelligent Transportation Systems Magazine* 9, 3 (2017), 100–113.

[20] Yang Li, Yangyan Li, Dimitrios Gunopulos, and Leonidas Guibas. 2016. Knowledge-based trajectory completion from sparse GPS samples. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* 33.

[21] He Liu, Yuan Gao, Lu Lu, Siyuan Liu, Lionel Ni, and Huamin Qu. 2011. Visual analysis of route diversity. *In Proceedings of the 2011 IEEE Conference on Visual Analytics Science and Technology*. 171–180.

[22] L. Liu, C. Andris, and C. Ratti. 2010. Uncovering cabdrivers' behaviour patterns from their digital traces. *Computers, Environment and Urban Systems* 34, 6 (2010), 541–548.

[23] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence*. 194–200.

[24] Zhicheng Liu, Fabio Miranda, Weiting Xiong, Junyan Yang, Qiao Wang, and Claudio Silva. 2020. Learning geo-contextual embeddings for commuting flow prediction. In In *Proceedings of the AAAI Conference on Artificial Intelligence*. 808–816.

[25] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 352–361.

[26] Massimiliano Luca, Gianni Barlacchi, Bruno Lepri, and Luca Pappalardo. 2021. A survey on deep learning for human mobility. *ACM Computing Surveys* 55, 1 (2021), 1–44.

[27] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods on Natural Language Processing*. 1412–1421.

[28] Jianming Lv, Qing Li, Qinghui Sun, and Xintong Wang. 2018. T-CONV: A convolutional neural network for multi-scale taxi trajectory prediction. In *2018 Proceedings of the IEEE International Conference on Big Data and Smart Computing*. 82–89.

[29] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. 2008. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM Symposium on Applied Computing*. 863–868.

[30] S. Phithakkitnukoon, T. Horanont, G. D. Lorenzo, R. Shibasaki, and C. Ratti. 2010. Activity-aware map: Identifying human daily activity pattern using mobile phone data. *Human Behavior Understanding* 6219 (2010), 14–25.
[31] Yan Qu and Jun Zhang. 2013. Trade area analysis using user generated mobile location data. In *WWW'13*. 1053–1064.
[32] Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part c: Emerging Technologies* 15, 5 (2007), 312–328.
[33] Mahmood Rahmani and Haris N. Koutsopoulos. 2013. Path inference from sparse floating car data for urban networks. *Transportation Research Part C: Emerging Technologies* 30 (2013), 41–54.
[34] C. Ratti, R. M. Pulselli, S. Williams, and D. Frenchman. 2006. Mobile landscapes: using location data from cellphones for urban analysis. *Environment and Planning B: Planning and Design* 33, 5 (2006), 727–748.
[35] Alberto Rossi, Gianni Barlacchi, Monica Bianchini, and Bruno Lepri. 2019. Modelling taxi drivers' behaviour for the next destination prediction. *IEEE Transactions on Intelligent Transportation Systems* 21, 7 (2019), 2980–2989.
[36] Irum Sanaullah, Mohammed Quddus, and Marcus Enoch. 2016. Developing travel time estimation methods using sparse GPS data. *Journal of Intelligent Transportation Systems* 20, 6 (2016), 532–544.
[37] Chaoming Song, Tal Koren, Pu Wang, and Albert-László Barabási. 2010. Modelling the scaling properties of human mobility. *Nature Physics* 6, 10 (2010), 818–823.
[38] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-Laszlo Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021.
[39] Junkai Sun, Junbo Zhang, Qiaofei Li, Xiuwen Yi, Yuxuan Liang, and Yu Zheng. 2020. Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[40] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 3104–3112.
[41] Jinjun Tang, Jian Liang, Tianjian Yu, Yong Xiong, and Guoliang Zeng. 2021. Trip destination prediction based on a deep integration network by fusing multiple features from taxi trajectories. *IET Intelligent Transport Systems* 15, 9 (2021), 1131–1141.
[42] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. 2009. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. 85–98.
[43] Martin Treiber and Arne Kesting. 2013. Trajectory and floating-car data. In *Proceedings of the Traffic Flow Dynamics*. 7–12.
[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*. 5998–6008.
[45] A Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13, 2 (1967), 260–269.
[46] Xingrui Wang, Xinyu Liu, Ziteng Lu, and Hanfang Yang. 2021. Large scale GPS trajectory generation using map based on two stage GAN. *Journal of Data Science* 19, 1 (2021), 126–141.
[47] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 25–34.
[48] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudre-Mauroux. 2020. Location prediction over sparse user mobility traces using RNNs: Flashback in hidden states!. In In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. 2184–2190.
[49] Xin Yao, Yong Gao, Di Zhu, Ed Manley, Jiaoe Wang, and Yu Liu. 2020. Spatial origin-destination flow imputation using graph convolutional networks. *IEEE Transactions on Intelligent Transportation Systems* (2020).
[50] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2013. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering* 25, 1 (2013), 220–232.
[51] Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Transactions on Intelligent Systems and Technology* 6, 3 (2015), 29.
[52] Ali Zonoozi, Jung-jae Kim, Xiao-Li Li, and Gao Cong. 2018. Periodic-CRN: A convolutional recurrent model for crowd density prediction with recurring periodic patterns. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3732–3738.

Received 11 April 2021; revised 25 July 2022; accepted 7 September 2022