Multi-Task Adversarial Learning for Semi-Supervised Trajectory-User Linking

Sen Zhang[1,2]*, Senzhang Wang[3,2]*, Xiang Wang[4]† (✉), Shigeng Zhang[3], Hao Miao[5], and Junxing Zhu[4]

[1] Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2] Shenzhen Research Institute, Nanjing University of Aeronautics and Astronautics, Shenzhen, China
[3] Central South University, Changsha, China
[4] National University of Defense Technology, Changsha, China
[5] Aalborg University, Aalborg, Denmark

zhangsen@nuaa.edu.cn, {szwang,sgzhang}@csu.edu.cn, haom@cs.aau.dk, {xiangwangcn,zhujunxing}@nudt.edu.cn

Abstract. Trajectory-User Linking (TUL), which aims to link the trajectories to the users who have generated them, is critically important to many real applications. Existing approaches generally consider TUL as a supervised learning problem which requires a large number of labeled trajectory-user pairs. However, in real scenarios users may not be willing to make their identities publicly available due to data privacy concerns, leading to the scarcity of labeled trajectory-user pairs. In addition, the trajectory data are usually sparse as users will not always check-in when they go to POIs. To address these issues, in this paper we propose a multi-task adversarial learning model named TULMAL for semi-supervised TUL with spare trajectory data. Specifically, TULMAL first conducts sparse trajectory completion through a proposed seq2seq model. Kalman filter is also coupled into the decoder of the seq2seq model to calibrate the generated new locations. The completed trajectories are next input into a generative adversarial learning model for semi-supervised TUL. The insight is that we consider all the users and their trajectories as a whole and perform TUL in the data distribution level. We first project users and trajectories into the common latent feature space through learning a projection function (generator) to minimize the distance between the user distribution and the trajectory distribution. Then each unlabeled trajectory will be linked to the user who is closest to it in the latent feature space without much guidance of labels. The two tasks are jointly conducted and optimized under a multi-task learning framework. Extensive experimental results on two real-world trajectory datasets demonstrate the superiority of our proposal by comparison with existing approaches.

Keywords: Trajectory completion · Trajectory-user linking · Adversarial learning · Multi-task learning.

* Equal contribution
† Corresponding author

1 Introduction

With the rapid development of satellite positioning technology and location-based services, many location-related mobile data such as human trajectory data and taxi OD data are ubiquitous nowadays. The large volume of human mobility data can facilitate us to have a better understanding on human behavior patterns and provide great opportunities for various trajectory mining tasks [11, 13, 33]. For example, the user mobility data collected from location-based social networks (LBSNs) such as Foursquare, can be used for POI recommendation [5, 17], trajectory classification [2, 4], traffic prediction [20–24, 29] and human mobility prediction [6, 9, 14].

In many online applications, users usually are not willing to make their personal identity information associated with their trajectories publicly available due to privacy concerns. In such a case the platforms can only collect the trajectory data, but the users who have generated them are unknown. Linking the trajectories to the users who have generated them, which is also called Trajectory User Linking (TUL), is fundamentally important to many tasks such as personalized POI recommendation and terrorists/criminal identification [7].

Existing works generally model TUL as a supervised learning task, which use RNN-based methods to learn a projection function between trajectories and users based on a large number of labeled trajectory-user pairs. TULER [7] is the first model proposed to address the TUL problem, which uses RNN to model the trajectory sequences and learn the dependencies between the location points to the users for TUL. Zhou et al. [33] proposed to use variational autoencoder to learn the hierarchical semantic features of user trajectories. The unlabeled data was also incorporated to deal with the data sparsity issue to improve TUL performance. DeepTUL [15] focused on the TUL task by learning the multi-periodic nature of user mobility from their historical trajectories and exploiting both spatial and temporal features of the trajectory data.

However, the performance of existing works may not be promising in real application scenarios due to the following two major challenges. First, in many LBSN platforms like Foursquare, users will not always check in and share their locations when going to a POI. It is common that users are not willing to check in due to data privacy concerns, which results in huge amount of sparse and incomplete trajectories. Existing works mostly consider the user trajectories are complete, and thus they may not be applicable in real applications. Second, existing supervised learning based methods need a large number of annotated trajectory-user pairs, which is extreamly time consuming and costly. How to conduct TUL with a few labeled trajectory-user pairs under a semi-supervised learning framework is challenging and less explored.

To address the above challenges, this paper proposes a multi-task adversarial learning model named TULMAL to perform sparse trajectory completion and semi-supervised TUL simultaneously. Specifically, TULMAL first completes the sparse raw trajectory data through a proposed seq2seq model. The sparse trajectory data are first input the encoder of the seq2seq model to learn the latent feature representations. Then motivated by the effectiveness of Kalman filter [10] in calibrating noise estimation for temporal data, we adopt Kalman filtering to calibrate the estimated new locations of the completed trajectory, which is coupled into the decoder of the seq2seq model. The completed trajectories are then input into an adversarial learning model for TUL. Instead of matching the trajectory-user pairs one by one, we consider all the users and trajectories as a whole and perform TUL from the data distribution level. We aim to learn a projection function Φ to embed users and trajectories into a common latent space. With the assumption that two users are similar if their trajectories are similar and vice versa, the projection Φ should make the trajectory close to the user who has generated it in the feature space. To this end, TULMAL uses an adversarial learning framework to learn the projection function Φ. Specifically, TULMAL contains an encoder E, a decoder O and a discriminator D. Encoder E maps the feature vectors of trajectories into a shared latent space, and decoder O projects the latent space features into the user space as the generated samples. The encoder and decoder together work as a projection function Φ. The discriminator D aims to distinguish the real instances of users from the samples generated by the decoders. Through adversarial learning, the discriminator essentially estimates the approximate Wasserstein distance between the user distribution and the projected trajectory distribution. Through the competition with discriminators, the projection function Φ will be updated to minimize the estimated Wasserstein distance. Given a new unlabeled trajectory, it will be projected into the user space by Φ first and then be linked to the user who is closest to it. In summary, our main contributions are as follows:

- To the best of our knowledge, we are the first to study the semi-supervised TUL problem with sparse and incomplete trajectory data.
- A novel model TULMAL is proposed to effectively address the studied problem. TULMAL first conducts sparse trajectory completion with a Kalman filter enhanced seq2seq model, and then performs semi-supervised TUL with an adversarial learning framework. The two tasks are jointly conducted under a multi-task learning framework.
- We conduct extensive experiments on two real-world datasets to evaluate the effectiveness of TULMAL. The experimental results show that our model provides significant performance improvement over existing state-of-the-arts.

2 Related Work

2.1 Trajectory Completion

Existing works for trajectory completion can be roughly categorized into the following two types. The first type of works is to directly complete the trajectories with missing locations, and the second type aims to recovery the trajectories through the next step or short-term POI prediction. For the direct location completion approach, Zheng et al. [31] proposed to infer the missing part of sparse trajectories by comparing the similarity of historical trajectories with the sparse trajectories. An attentional neural network model AttnMove [26] was proposed to complete individual trajectories by recovering unobserved locations based on historical trajectories. Xi et al. [25] proposed a bidirectional spatial and temporal dependency and a dynamic preference model of users to identify missing POI check-ins. MTrajRec [16] used a seq2seq multi-task learning model to patch missing trajectory points while mapping them to the road network.

For next location prediction works, STRNN proposed by Liu et al. [12] tried to model the temporal and spatial context of each layer. Specifically, STRNN used a specific excess matrix for different time intervals and geographical distances to predict the next POI. Feng et al. [3] proposed a recurrent neural network with multimodal embedding named DeepMove to capture the complex sequential transitions by jointly embedding multiple factors that governed human movement. DeepMove also used a historical attention model with two mechanisms to capture multi-level periodicity, effectively exploiting the nature of periodicity to enhance recurrent neural networks' mobility prediction. However, a significant drawback of existing works is that they are not effective to reduce data noise in trajectory data completion.

2.2 Trajectory-User Linking

Existing TUL models are mostly supervised, which use machine learning models especially RNN-based approaches to learn a projection function between users and trajectories through a large number of labeled trajectory-user pairs. TULER [7] is the first model proposed for TUL. It used RNN to model the trajectory sequence for capturing the dependencies between location points. DeepTUL [15] learned the multi-periodic nature of user mobility from the user's historical trajectory and used both spatial and temporal features of the trajectory data for user-trajectory matching. Considering the large number of users, TULSN [28] was proposed to model the trajectory data by linking networks, and only a small amount of trajectory data were needed for training the model. TULVAE [33] was a novel semi-supervised variational autoencoder framework, which used variational autoencoder to learn the hierarchical semantic features of user trajectories and incorporated unlabeled data to solve the data sparsity problem for TUL. However, the data sparsity issue in many real scenarios is largely ignored by existing works. Existing works usually need a large number of annotated trajectory-user pairs, which is labor intensive and costly thus infeasible in many applications.

3 Problem Definition

In this section, we will first give definitions of some terminologies, and then formally define the studied problem.

Definition 1. *(Cell region). We divide a city under study into a set of equal-sized grid cells, denoted as R. Each cell region $r \in R$ is a square region. The coordinates of a cell region r are denoted by its latitude and longitude $\langle x_r, y_r \rangle$.*

Definition 2. *(Trajectory). A trajectory $\widetilde{T} = \langle s_1, s_2, \cdots, s_n \rangle$ is defined as a sequence of geographically located points with time order, where $s_i = \langle x, y, t, r \rangle$ represents a location point consisting of latitude x, longitude y, timestamp t and the cell region r where s_i is located.*

Note that $\widetilde{T}$ is sparse with some locations missing. We denote the complete trajectory as $T = (p_1, p_2, ..., p_n)$, where T includes all visited locations and $\widetilde{T} \in T$. The sparse trajectory dataset $\widetilde{\mathbf{T}} = \left\{ \widetilde{T}_1, \widetilde{T}_2, \cdots, \widetilde{T}_m \right\}$ contains a small number of labeled or linked trajectories $\widetilde{\mathbf{T}}^l$ and a large number of unlabeled or unlinked ones $\widetilde{\mathbf{T}}^u$. Let $\mathbf{U} = \{u_1, u_2, \cdots, u_n\}$ denote the user set. We assume that each user has some linked trajectories, and the linked trajectory set associated with each user is denoted as $\widetilde{\mathbf{T}}^l = \left\{ (\widetilde{T}^{u_1}, u_1), (\widetilde{T}^{u_2}, u_2), \cdots, (\widetilde{T}^{u_n}, u_n) \right\}$, where $(\widetilde{T}^{u_i}, u_i)$ means trajectories $\widetilde{T}^{u_i}$ belongs to u_i, and thus $\widetilde{T}^{u_i}$ can be used to represent u_i. The studied problem is formally defined as follows.

Problem Statement. Given the sparse trajectory set $\widetilde{\mathbf{T}}$, the user set $\mathbf{U}$ and some trajectory-user pairs $\widetilde{\mathbf{T}}^l = \left\{ (\widetilde{T}^{u_1}, u_1), (\widetilde{T}^{u_2}, u_2), \cdots, (\widetilde{T}^{u_n}, u_n) \right\}$, our goal is to complete the trajectories in $\widetilde{\mathbf{T}}$ to obtain the complete trajectories $\mathbf{T} = \{\mathbf{T}^l, \mathbf{T}^u\}$, and then learn a projection function Φ to link all the trajectories in $\mathbf{T}^u$ to the users in $\mathbf{U}$.

4 Methodology

Fig. 1 shows the model framework, which contains the trajectory completion step and the adversarial learning based TUL step. In the first step, we propose SeqKF model which integrates the Seq2Seq model with Kalman filter to accomplish the trajectory completion task. We first obtain the cell region where the coordinates are located by Seq2Seq. Then Kalman filtering is used for fine-grained calibration to obtain the exact coordinate values. In the adversarial learning step, we aim to learn a projection function that minimizes the distance between the generated trajectory distribution and the user distribution.

4.1 SeqKF for Trajectory Completion

The proposed SeqKF model for trajectory completion consists of the encoder and the decoder with Kalman filter. The encoder learns the spatio-temporal dependency of the trajectories, while the decoder generates the completed trajectory. Inspired by [32], instead of directly predicting the coordinate values of the missing trajectory points, we predict the grid cells where it is located. This approach allows for easier modeling than using coordinate values directly. Then we predict the cell region where the location point is located by the Seq2Seq model and use the center of the cell as the predicted coordinate value of the location point. Finally, we correct the predicted coordinate values by a Kalman filter to obtain an accurate prediction.

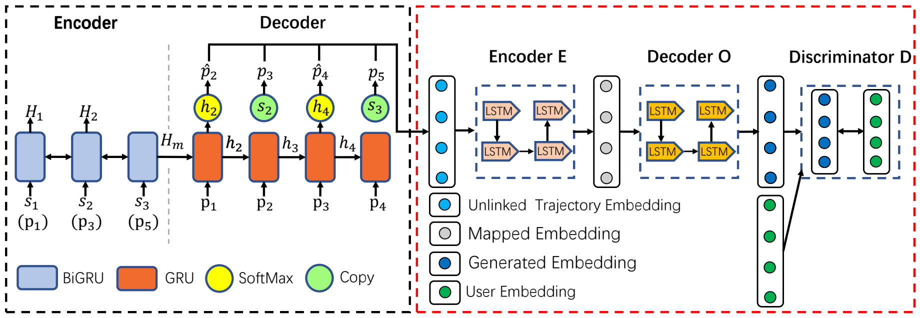


Fig. 1. The TULMAL model. The SeqKF step is in the black dashed box, and the adversarial learning based TUL step is in the red dashed box.

Encoder. We encode the input sparse trajectory as a fixed vector and feed it into the Bidirectional Gated Recurrent Unit (BiGRU). GRU is a variant of Long Short Term Memory (LSTM) network, which is able to learn long-term dependencies of continuous data without performance degradation, while BiGRU can capture both forward and backward temporal dependencies.

BiGRU is actually two GRUs processing the data into forward and backward paths, and then combining the outputs in these two directions to obtain the final hidden state. The forward and backward hidden states in time step i are denoted as $\overrightarrow{H_i}$ and $\overleftarrow{H_i}$. Then the output H_i in time step i is the combination of $\overrightarrow{H_i}$ and $\overleftarrow{H_i}$, i.e. $H_i = \overrightarrow{H_i} + \overleftarrow{H_i}$.

Decoder with Kalman filter. The GRU decoder is used to recover the sparse trajectories $\widetilde{\mathbf{T}} = \langle s_1, s_2, \cdots, s_m \rangle$. Unlike the standard Seq2Seq, in our model, for the points presenting in the raw trajectory, we use the idea of replication operation that is widely used in NLP tasks [8,27]. We replicate these points directly from the output slot of the decoder as follows

$$p_i = \begin{cases} \hat{p}_i, \text{ if } j_k < i < j_{k+1}, \\ s_k, \text{ if } i = j_k, \end{cases} \tag{1}$$

where $\hat{p}_i$ represents the cell where the missing points are predicted by decoder.

To take global relevance into account, we add an attention mechanism so that the hidden unit h_i in the decoder is updated by

$$h_i = GRU(h_{i-1}, p_{i-1}, a_i, H_m), \tag{2}$$

where H_m is the output state from the encoder and a_i is the weighted sum computed from all output vectors H in the encoder, which is expressed as

$$\begin{aligned} a_i &= \sum_{i=1}^{m} \alpha_{i,k} H_i, \\ \alpha_{i,k} &= \frac{exp(u_{i,k})}{\sum_{k'=1}^{m} exp(u_{i,k'})}, \\ u_{i,k} &= v^T \cdot tanh(W_h h_i + W_H H_k), \end{aligned} \tag{3}$$

where v, W_h and W_H are learnable parameters and h_i denotes the current state of the decoder. When we obtain the hidden state h_i from the decoder, for points that are not in the trajectory, we apply the softmax function to generate the corresponding cells of the missing trajectory points conditional on the probability of $p(c|h_i)$ as

$$pro(c|h_i) = \frac{exp({h_i}^T \cdot w_c)}{\sum_{c' \in C} exp({h_i}^T \cdot w_{c'})}, \tag{4}$$

where w_c is the c-th column vector of the trainable parameter matrix $\mathbf{W}_c$.

Now we have the cell regions corresponding to the missing locations, we next combine the Kalman filter (KF) with the decoder to estimate the exact locations. KF is essentially an optimal state estimator under the assumption of linear and Gaussian noise. It is used to calibrate the coarse-grained predictions generated by the Seq2Seq output. In the KF model, we denote the state of the object at timestep k as g_k, which is denoted as

$$g_k = \mathbf{A} g_{k-1} + d_g, \quad d_g \sim \mathbf{N}\left(0, \mathbf{P}\right), \tag{5}$$

where $\mathbf{A}$ is the state update matrix, d_g denotes Gaussian noise, and $\mathbf{P}$ denotes the covariance matrix of d_g. The current state can be obtained from the measurement value z_k, which is denoted as

$$z_k = \mathbf{B} g_k + e_z, \quad e_z \sim \mathbf{N}\left(0, \mathbf{Q}\right), \tag{6}$$

where $\mathbf{B}$ is the measurement matrix, e_z denotes the measurement Gaussian noise, and $\mathbf{Q}$ denotes the covariance matrix of e_z. The KF model mainly estimates the true state value g based on the predicted value $\hat{g}^-$ and the measured value z, and it is divided into two steps: prediction and calibration.

In the prediction phase, the KF model predicts the prior value $\hat{g}^-$ and the prior error covariance matrix $\mathbf{R}^-$ at time step k by the following equations

$$\hat{g}_k^- = \mathbf{A} \hat{g}_{k-1}, \tag{7}$$

$$\mathbf{R}_\mathbf{k}^- = \mathbf{A} \mathbf{R}_\mathbf{k-1} \mathbf{A}^T + \mathbf{P}. \tag{8}$$

In the calibration phase, the KF model obtains the posteriori estimated state value $\hat{g}$ and updates the covariance matrix $\mathbf{R}$ by using the measured value z and

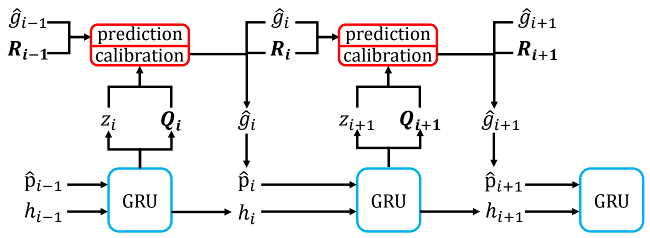


Fig. 2. Details of the combination of the decoder and Kalman filter, where the red part shows the two steps of updating and calibration of the Kalman filter.

the a priori value $\hat{g}^-$ by the following equations

$$\hat{g}_k = \hat{g}_k^- + K_k \left(z_k - \mathbf{B}\hat{g}_k^-\right), \tag{9}$$

$$\mathbf{R_k} = \left(\mathbf{I} - K_k\mathbf{B}\right)\mathbf{R_k^-}, \tag{10}$$

where K_k is the optimal Kalman gain, which combines the priori value $\hat{g}^-$ and the measured value z for final estimation. K_k is given by the following equation

$$K_k = \mathbf{R_k^-}\mathbf{B^T}\left(\mathbf{B}\mathbf{R_k^-}\mathbf{B^T} + \mathbf{Q}\right)^{-1}. \tag{11}$$

K_k is used to denote the importance of the estimation error covariance matrix $\mathbf{R_k}$ and the measurement error covariance matrix $\mathbf{Q}$.

KF has two inputs z and $\mathbf{Q}$. For the measurement z, we take out the center coordinates (x_{c_i}, y_{c_i}) of the prediction unit c_i output from decoder as the value of z_i. For $\mathbf{Q}$, the traditional KF model sets $\mathbf{Q}$ as a fixed prior parameter, but intuitively, the uncertainty of the measurements should be constantly changing at different time periods. Therefore, we introduce a dynamic covariance matrix. For a given set of grid cells R, we aggregate the central coordinates of all grids r into a matrix $\mathbf{V}$ of size $2 * |R|$. At each timestamp i, we calculate the currently estimated expected coordinate vector by

$$\bar{v}_i = \sum_{c' \in G} pro\left(c'|h_i\right) \cdot v_{c'}, \tag{12}$$

where $pro\left(c'|h_i\right)$ is the predicted probability of c' calculated by the softmax function in Eq. (4), and $u_{c'}$ represents the c-th column vector in the matrix $\mathbf{V}$. We then calculate the covariance matrix $\mathbf{Q}$ by

$$\mathbf{Q_i} = \sum_{r' \in R} pro\left(r'|h_i\right) \cdot \left(v_{r'} - \bar{v}_i\right) \cdot \left(v_{r'} - \bar{v}_i\right)^T, \tag{13}$$

where $\{pro\left(r'|h_i\right)\}_{r' \in R}$ is used to combine the covariance matrix of each cell as the expected covariance of the measurement z.

As shown in Fig. 2, at each timestep i, the decoder cell feeds the center coordinates of the predicted cell into the KF component, and KF calibrates the observation z_i through two procedures, prediction and correction, to obtain the final prediction g_i. Then g_i is further discredited into a grid cell $\hat{p}_i$, which is used as the input of the next decoder cell. By this combination, the prediction noise can be effectively reduced.

Loss Function Given a training set $\mathbf{D} = \left\{\mathbf{T}, \widetilde{\mathbf{T}}\right\}$ containing a set of sparse trajectories $\widetilde{\mathbf{T}}$ and the corresponding completed trajectories $\mathbf{T}$, we use the cross entropy as the loss function.

$$L_1 = \sum_{(\mathbf{T},\widetilde{\mathbf{T}})\in\mathbf{D}} -log(pro(\mathbf{T}|\widetilde{\mathbf{T}})). \tag{14}$$

To optimize the KF component, we use the mean squared error as the loss function, which is defined as

$$L_2 = \frac{1}{2} \sum_{(\mathbf{T},\widetilde{\mathbf{T}})\in\mathbf{D}} \sum_{p_i\in\mathbf{T}\ and\ p_i\notin\widetilde{T}} \left(\begin{bmatrix} p_i.x \\ p_i.y \end{bmatrix} - \hat{g}_i \right)^2 . \tag{15}$$

Then the loss function of the trajectory completion task can be expressed as

$$L_{coml} = L_1 + \lambda L_2, \tag{16}$$

where λ is a parameter to balance the importance of the two terms.

4.2 Adversarial Learning for Semi-Supervised TUL

Generator The generator aims to generate the representations of the trajectories from the raw feature space to the user space, and it consists of an encoder and a decoder. The encoder is responsible for mapping the input trajectories into a latent space, and the decoder is responsible for projecting the latent embedding in the latent space into the target user space. We use LSTM as the encoder and decoder. After mapping the trajectories to the target user space by the generator, we can identify the real instances from the users by the discriminator D. Next we will derive the discriminator needed for the TUL task starting from the objective function.

Objective Function Given the distribution $\mathbb{D}^{\mathbf{T}}$ represented by the set of trajectories and the distribution $\mathbb{D}^{\mathbf{U}}$ represented by the set of users, the objective of TULMAL can be defined as follows:

$$\underset{\Phi}{min} WD(\mathbb{D}^{\mathbf{U}}, \mathbb{D}^{\Phi(\mathbf{T})}) = \underset{\gamma\in\Upsilon(\mathbb{D}^{\mathbf{U}},\mathbb{D}^{\Phi(\mathbf{T})})}{inf} \mathbb{E}_{(\mathbf{U},\Phi(\mathbf{T}))\sim\gamma}[d(\mathbf{U}, \Phi(\mathbf{T}))]. \tag{17}$$

The right-hand side of Eq. (17) is a representation of the Wasserstein distance, which measures the distance between two probability distributions $\mathbb{D}^{\mathbf{U}}$ and $\mathbb{D}^{\Phi(\mathbf{T})}$. $\Upsilon(\mathbb{D}^{\mathbf{U}}, \mathbb{D}^{\Phi(\mathbf{T})})$ is the set of all possible joint probability distributions for the combination of distributions $\mathbb{D}^{\mathbf{U}}$ and $\mathbb{D}^{\Phi(\mathbf{T})}$. d represents the distance between two points (set as Euclidean distance in this paper). WD aims to find the ideal joint distribution Υ to reach the expectation infimum. However, it is difficult to compute $inf_{\gamma \in \Upsilon(\mathbb{D}^{\mathbf{U}}, \mathbb{D}^{\Phi(\mathbf{T})})}$ [30] by traversing all joint distributions. The work [18] presents a simple version of WD, which can be formulated as follows when Kantorovich-Rubinstein duality is satisfied:

$$WD = \frac{1}{K} \underset{\|f\|_L \leq K}{sup} \mathbb{E}_{\mathbf{U} \sim \mathbb{D}^{\mathbf{U}}} f(\mathbf{U}) - \mathbb{E}_{\Phi(\mathbf{T}) \sim \mathbb{D}^{\Phi(T)}} f(\Phi(\mathbf{T})). \tag{18}$$

where the function f is required to be K-Lipschitz continuous. For Eq. (18), we have to learn an ideal K-Lipschitz function f to implement it. Since the neural network itself has a powerful approximation capability, we choose a multilayer feedforward network to find f. It can be regarded as a discriminator D that distinguishes between the target and generated samples, and the loss of the discriminator can then be expressed as follows:

$$\underset{\alpha}{min}\ L_D = \mathbb{E}_{\Phi(\mathbf{T}) \sim \mathbb{D}^{\Phi(T)}} D(\Phi(\mathbf{T})) - \mathbb{E}_{\mathbf{U} \sim \mathbb{D}^{\mathbf{U}}} D(\mathbf{U}), \tag{19}$$

where α is the set of parameters of the feedforward network f (i.e., the discriminator D). To satisfy the K-Lipschitz restriction, we use the clipping trick by sandwiching the weights α in a small window [-c,c] after each gradient update.

The generator Φ is designed to minimize WD. For Eq. (19), Φ exists only in the second term on the left-hand side of the equation, so we can learn the ideal Φ by minimizing the following loss:

$$\underset{\Phi}{min}\ L_{\Phi} = \mathbb{E}_{\Phi(\mathbf{T}) \sim \mathbb{D}^{\Phi(\mathbf{T})}} D(\Phi(\mathbf{T})). \tag{20}$$

As the loss of the generator Φ gradually decreases, the loss of the discriminator D, i.e., WD, also decreases, so that trajectories belonging to the same user are grouped together in the latent space. Meanwhile, we also incorporate a small number of annotations $\mathbf{T}^l$. For a matched pair of trajectories (T^{u_i}, u_i) in $\mathbf{T}^l$, our goal is to minimize the distance between the trajectory and the user as follows:

$$\underset{\Phi}{min}\ L_W = \frac{\theta_w}{|T^l|} \sum_{(\mathbb{T}^{u_i}, u_i) \in \mathbb{T}^l} dis(\Phi(T^{u_i}), u_i), \tag{21}$$

where θ_w is the hyper-parameter controlling the weight of the loss L_W.

Adversarial Loss. The loss function for adversarial learning is as follows

$$L_{al} = L_{\Phi} + L_D + L_W, \tag{22}$$

where L_{Φ} represents the loss of generator Φ, L_D represents the loss of discriminator D, and L_W represents the loss of labeled trajectory-user pairs.

4.3 Final Objective Function

The sparse trajectory completion and TUL tasks are optimized simultaneously, and the overall loss L of the two tasks is as follows

$$L = L_{al} + \mu L_{coml}, \tag{23}$$

Table 1. Dataset Description.$|U|$: number of users; $|T_r|$ / $|T_{te}|$: number of trajectories for training and testing; $|P|$: number of trajectory point; $|R|$: average length of trajectories (before segmentation).

Dataset	$\|U\|$	$\|T_{tr}\|$ / $\|T_{te}\|$	$\|P\|$	$\|R\|$
NYC	113	9122/2280	3561	214
TKY	258	15843/3871	4126	188

where L_{coml} is the loss for trajectory completion, μ is the hyperparameter, and L_{al} is the loss for TUL. The work is implemented with the Huawei MindSpore AI computing framework.

5 Experiment

5.1 Dataset and Experiment Setup

Dataset. Two datasets collected from Foursquare are used in our experiment. **NYC dataset** records about 10 months of check-ins in New York City. Each check-in includes its timestamp, GPS coordinates and semantic information (represented by fine-grained venue-categories). **TKY dataset** contains about ten months of check-in records in Tokyo.

Following [34], we randomly select $|U|$ users and their generated trajectories from both datasets for evaluation. For each trajectory, we randomly sample $r\%$ points and remove the others to simulate the sparse trajectory. In our experiments we keep 50% and 70% points for each trajectory, respectively. 10% of the entire trajectories and their users are used as annotations for supervision. Table 1 shows the details of the two datasets.

Evaluation metrics. $ACC@K$ and $macro\text{-}F1$ are used to evaluate the performance of the model. In addition, to verify the effectiveness of trajectory completion, we use three metrics $RMSE$, $NDTW$, and EDR for evaluation. $ACC@K$ is to evaluate the accuracy of the TUL problem, which is defined as

$$ACC@K = \frac{correctly\ linked\ trajectories@K}{the\ number\ of\ trajectories}.$$

$Macro\text{-}F1$ is the harmonic mean of accuracy ($macro\text{-}P$) and recall ($macro\text{-}R$), averaged over all categories (users in TUL).

$$macro\text{-}F1 = \frac{2 \times macro\text{-}P \times marco\text{-}R}{marco\text{-}P + macro\text{-}R}$$

$RMSE$ is the root mean square error between the actual and predicted values of the coordinates of the missing trajectory points that is formulated as

$$RMSE = \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(dis\left(p_j, \hat{p}_j\right)\right)^2},$$

where $dis\left(p_j, \hat{p_j}\right)$ represents the Euclidean distance between the true value p_j and the predicted value $\hat{p}_j$.

$NDTW$ is the normalized dynamic time warping distance between two trajectories, which is modified from dynamic time warping distance (DTW),

$$NDTW\left(T, \widetilde{T}\right) = \frac{DTW\left(T, \widetilde{T}\right)}{length\left(T\right)}.$$

EDR is the edit distance on real sequence. Specifically, given two trajectories T and $\widetilde{T}$, the edit distance between them is the number of operations required to transform T into $\widetilde{T}$ through insert, delete and replace operations.

Baselines. To evaluate the effectiveness of TULMAL, we first compare it with the following baseline methods: DTW [19] and LCSS [1], which compute the distance between two trajectories by using different criteria for comparing similarity; TULER [7], TULER-LSTM, TULER-GRU, BiTULER, TULVAE [34] and DeepTUL [15], which use deep learning classification models to learn the projection function between trajectories and users.

To further test the power of adversarial learning in the TUL problem, we compare TULMAL with its variant model TULMAL-NoSeqKF, which removes the completion module. We also compare TULMAL with the baseline methods plus our proposed trajectory completion model SeqKF to study the effectiveness of SeqKF. We also select two trajectory completion baseline methods, DeepMove [3] and STRNN [12], to compare with SeqKF to verify its effectiveness in trajectory completion.

5.2 Experimental Results

Tables 2 and 3 show the performance of various methods on the two datasets in terms of $ACC@K$ and $macro$-$F1$. Table 2 shows the comparison result at a sampling rate of 70% and Table 3 shows the comparison result at a sampling rate of 50%, where the best values are highlighted in bold. In Table 2, one can observe that on the dataset NYC, TULMAL-NoSeqKF improves by 2.86%, 3.15% and 1.79% in terms of $ACC@1$, $ACC@5$ and $macro$-$F1$, respectively, compared to the best baseline method DeepTUL. This superior result is due to its ability to exploit the multi-period nature of user mobility and to address the data sparsity issue. After adding SeqKF to the baseline methods, the performance of all the methods improve, which indicates that the trajectory completion component is effective to improve the TUL performance. Our method TULML achieves the best performance, improving the three metrics by 4.94%, 3.51% and 2.72%, respectively, compared with the best baseline method SeqKF+DeepTUL. This indicates that our adversarial learning based TUL component is more effective than the baselines.

In Table 3 one can see that TULML also achieves the best performance when the sampling rate is 50%. Compared with DeepTUL, TULMAL-NoSeqKF improves by 2.39%, 4.9% and 2.35% in terms of the three metrics, respectively. Compared with SeqKF+DeepTUL, TULMAL improves the three metrics by

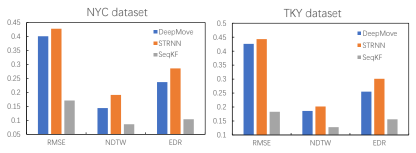
Table 2. Performance comparison over two datasets under a sampling rate (SR) of 70%

	NYC			TKY		
	acc@1	acc@5	macro-F1	acc@1	acc@5	macro-F1
	u=113, SR=70%			u=258,SR=70%		
DTW	13.74%	24.58%	6.14%	10.08%	20.65%	5.41%
LCSS	15.21%	28.63%	7.54%	12.38%	23.86%	6.44%
TULER-LSTM	19.31%	46.65%	11.99%	16.14%	38.42%	9.18%
TULER-GRU	20.25%	46.82%	13.87%	17.31%	39.41%	9.68%
BiTULER	22.63%	50.42%	16.55%	18.52%	42.35%	12.94%
TULVAE	23.32%	50.25%	16.83%	18.65%	43.28%	13.76%
DeepTUL	25.62%	53.16%	19.00%	21.44%	45.83%	15.07%
TULMAL-NoSeqKF	28.48%	56.31%	20.79%	23.83%	48.11%	15.97%
SeqKF+DTW	15.43%	30.74%	7.62%	12.62%	25.83%	6.28%
SeqKF+LCSS	20.12%	37.57%	9.45%	18.07%	31.46%	8.10%
SeqKF+TULER-LSTM	23.37%	48.63%	16.20%	20.75%	42.96%	12.42%
SeqKF+TULER-GRU	25.14%	48.86%	16.40%	22.43%	42.75%	12.95%
SeqKF+BiTULER	26.84%	52.49%	17.69%	23.87%	45.62%	15.05%
SeqKF+TULVAE	27.89%	53.78%	19.28%	25.16%	48.37%	16.77%
SeqKF+DeepTUL	30.48%	57.86%	21.18%	27.45%	50.38%	18.78%
TULMAL	**35.42%**	**61.37%**	**23.90%**	**32.84%**	**51.48%**	**22.58%**

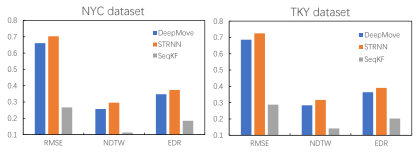
Table 3. Performance comparison over two datasets under a sampling rate (SR) of 50%

	NYC			TKY		
	acc@1	acc@5	macro-F1	acc@1	acc@5	macro-F1
	u=113,SR=50%			u=258,SR=50%		
DTW	9.58%	16.83%	4.43%	7.62%	13.27%	3.19%
LCSS	11.32%	22.67%	6.17%	8.51%	18.47%	4.62%
TULER-LSTM	15.38%	26.77%	7.65%	12.85%	22.66%	6.41%
TULER-GRU	16.19%	30.68%	8.74%	14.67%	25.14%	6.97%
BiTULER	17.84%	32.73%	9.42%	15.98%	27.31%	7.39%
TULVAE	19.18%	35.96%	11.37%	17.22%	30.37%	8.14%
DeepTUL	22.47%	41.85%	13.21%	20.05%	34.99%	11.65%
TULMAL-NoSeqKF	24.86%	46.75%	15.56%	21.96%	38.41%	12.87%
SeqKF+DTW	12.54%	21.48%	6.21%	9.63%	15.85%	4.31%
SeqKF+LCSS	13.79%	25.31%	7.34%	10.95%	21.07%	5.56%
SeqKF+TULER-LSTM	18.46%	31.24%	9.17%	14.16%	27.60%	7.47%
SeqKF+TULER-GRU	20.15%	34.58%	10.83%	16.52%	29.89%	8.51%
SeqKF+BiTULER	20.97%	36.27%	11.68%	17.83%	31.04%	9.33%
SeqKF+TULVAE	22.49%	40.62%	13.15%	19.57%	34.41%	11.60%
SeqKF+DeepTUL	25.91%	47.36%	15.91%	23.12%	38.77%	12.84%
TULMAL	**30.03%**	**53.64%**	**19.58%**	**26.70%**	**43.41%**	**15.49%**

4.12%, 6.28% and 3.67%, respectively. From the two tables one can also see that the results are generally better on the NYC dataset than that on the TKY



(a) 70% sampling rate



(b) 50% sampling rate

Fig. 3. Performance comparison on SeqKF at sampling rates of 70% and 50%

dataset. This is mainly because there are fewer users in NYC than in TKY, resulting in denser trajectories in NYC than in TKY. Denser trajectories provide more information and thus the models can achieve better performance.

5.3 Effectiveness of Trajectory Completion

Fig. 3 shows the performance comparison of the proposed SeqKF with other baseline methods under the sampling rates of 70% and 50%, respectively. In Fig. 3(a) one can observe that DeepMove is better than STRNN because it considers more history information. SeqKF model is better than DeepMove. This is because SeqKF can effectively reduce the effect of noise by using Kalman filter. One can also observe that the performance of the methods on NYC is better than that on TKY. This is mainly because the trajectories in TKY are sparser than those in NYC. In Fig. 3(b), SeqKF also significantly outperforms the two baselines DeepMove and STRNN. One can also see that the performance drop of SeqKF is much smaller than the other two methods when the sparsity rate changes from 70% to 50%. It further verifies SeqKF can effectively reduce the effect of noise and improve the robustness of the model by combining Seq2Seq with Kalman filter. The performance of the two baselines degrades quickly because the smaller sparsity rate leads to sparser trajectories.

5.4 Parameter Sensitivity Study

We finally investigate the performance sensitivity of TULMAL on three parameters: the weight μ of the final loss function, the annotation guidance weight θ in TUL, and the weight λ of the loss function in SeqKF. We let μ increase from 0 to 0.5, θ increase from 0.1 to 0.5, and λ increase from 0 to 0.25. In our experiment, the sampling rate is set to 70% for both datasets. As can be seen in Fig. 4, the performance of the model first increases and then decreases as μ keeps increasing, which indicates that an appropriate SeqKF loss can improve the performance of TUL, but a too large SeqKF loss can overwhelm the adversarial loss and thus hurt the performance. Similar result is produced for the annotation guidance weight θ. The performance of the model increases first and then decreases as θ increases. θ=0.3 is a suitable setting for both datasets. As λ increases, the RMSE first decreases and then increases. The best performance of

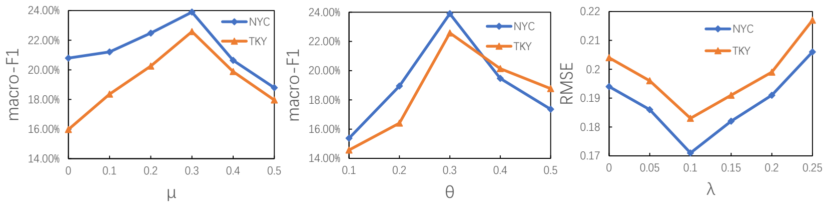


Fig. 4. The effect of three parameters μ, θ and λ on the model performance.

SeqKF is achieved when λ is 0.1, which proves that KF is effective in improving the performance of trajectory completion.

6 Conclusion

In this paper, we proposed a multi-task adversarial learning model for semi-supervised TUL with sparse trajectory data. TULMAL first used the trajectory completion component SeqKF to effectively complete the sparse trajectories. SeqKF combined Seq2Seq model and Kalman filter effectively to alleviate the noise for data completion. Then the TUL problem was solved by capturing the multi-periodicity of user movement through a proposed adversarial learning model. As TUL was conducted in the data distribution level rather than the trajectory-user pair data instance level, the proposed model required only a small number of annotations. We conducted extensive experiments on two real datasets. The experimental results showed that our proposal outperformed previous approaches in the two tasks.

7 Acknowledgments

This research was funded by National Science Foundation of China (No. 62172443 and 61802424), Guangdong Basic and Applied Basic Research Foundation (No.20 21A1515012239), CAAI-Huawei MindSpore Open Fund and the Fundamental Research Funds for the Central Universities (No.: NZ2020014).

References

1. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Proceedings of KDD workshop (1994)
2. Damiani, M.L., Guting, R.H.: Semantic trajectories and beyond. In: Proceedings of MDM (2014)
3. Feng, J., Li, Y., Zhang, C., Sun, F., Meng, F., Guo, A., Jin, D.: Deepmove: Predicting human mobility with attentional recurrent networks. In: Proceedings of WWW (2018)
4. Freitas, N., Silva, T., Macˆedo, J., Junior, L.M., Cordeiro, M.: Using deep learning for trajectory classification. In: Proceedings of ICAART (2021)

5. Gao, Q., Trajcevski, G., Zhou, F., Zhang, K., Zhong, T., Zhang, F.: Deeptrip: Adversarially understanding human mobility for trip recommendation. In: Proceedings of ACM SIGSPATIAL (2019)
6. Gao, Q., Zhou, F., Trajcevski, G., Zhang, K., Zhong, T., Zhang, F.: Predicting human mobility via variational attention. In: Proceedings of WWW (2019)
7. Gao, Q., Zhou, F., Zhang, K., Trajcevski, G., Luo, X., Zhang, F.: Identifying human mobility via trajectory embeddings. In: Proceedings of IJCAI (2017)
8. Gu, J., Lu, Z., Li, H., Li, V.O.: Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393 (2016)
9. Huang, L., Yang, Y., Chen, H., Zhang, Y., Wang, Z., He, L.: Context-aware road travel time estimation by coupled tensor decomposition based on trajectory data. Knowledge-Based Systems 245, 108596 (2022)
10. Kalman, R.E.: A new approach to linear filtering and prediction problems. Fluids Engineering 82D, 35–45 (1959)
11. Li, J., Wang, S., Zhang, J., Miao, H., Zhang, J., Yu, P.: Fine-grained urban flow inference with incomplete data. IEEE Transactions on Knowledge and Data Engineering pp. 1–14 (2022)
12. Liu, Q., Wu, S., Wang, L., Tan, T.: Predicting the next location: A recurrent model with spatial and temporal contexts. In: Proceedings of AAAI (2016)
13. Mahajan, R., Mansotra, V.: Predicting geolocation of tweets: using combination of cnn and bilstm. Data Science and Engineering 6(4), 402–410 (2021)
14. Miao, C., Luo, Z., Zeng, F., Wang, J.: Predicting human mobility via attentive convolutional network. In: Proceedings of WSDM (2020)
15. Miao, C., Wang, J., Yu, H., Zhang, W., Qi, Y.: Trajectory-user linking with attentive recurrent network. In: Proceedings of AAMAS (2020)
16. Ren, H., Ruan, S., Li, Y., Bao, J., Meng, C., Li, R., Zheng, Y.: Mtrajrec: Map-constrained trajectory recovery via seq2seq multi-task learning. In: Proceedings of ACM SIGKDD (2021)
17. Sun, K., Qian, T., Chen, T., Liang, Y., Nguyen, Q.V.H., Yin, H.: Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In: Proceedings of AAAI (2020)
18. Villani, C.: Optimal transport: old and new. Springer (2009)
19. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. In: Proceedings of ICDE (2002)
20. Wang, S., Miao, H., Chen, H., Huang, Z.: Multi-task adversarial spatial-temporal networks for crowd flow prediction. In: Proceedings of CIKM (2020)
21. Wang, S., Miao, H., Li, J., Cao, J.: Spatio-temporal knowledge transfer for urban crowd flow prediction via deep attentive adaptation networks. IEEE Transactions on Intelligent Transportation Systems 23(5), 4695–4705 (2021)
22. Wang, S., Zhang, J., Li, J., Miao, H., Cao, J.: Traffic accident risk prediction via multi-view multi-task spatio-temporal networks. IEEE Transactions on Knowledge and Data Engineering pp. 1–14 (2021)
23. Wang, S., Zhang, M., Miao, H., Peng, Z., Yu, P.S.: Multivariate correlation-aware spatio-temporal graph convolutional networks for multi-scale traffic prediction. ACM Transactions on Intelligent Systems and Technology 13(3), 1–22 (2022)
24. Wang, S., Zhang, M., Miao, H., Yu, P.S.: Mt-stnets: Multi-task spatial-temporal networks for multi-scale traffic prediction. In: Proceedings of SDM (2021)
25. Xi, D., Zhuang, F., Liu, Y., Gu, J., Xiong, H., He, Q.: Modelling of bi-directional spatio-temporal dependence and users' dynamic preferences for missing poi check-in identification. In: Proceedings of AAAI (2019)

26. Xia, T., Qi, Y., Feng, J., Xu, F., Sun, F., Guo, D., Li, Y.: Attnmove: History enhanced trajectory recovery via attentional network. arXiv preprint arXiv:2101.00646 (2021)
27. Yang, Z., Ma, J., Chen, H., Zhang, J., Chang, Y.: Context-aware attentive multilevel feature fusion for named entity recognition. IEEE Transactions on Neural Networks and Learning Systems pp. 1–12 (2022)
28. Yu, Y., Tang, H., Wang, F., Wu, L., Qian, T., Sun, T., Xu, Y.: Tulsn: Siamese network for trajectory-user linking. In: Proceedings of IJCNN (2020)
29. Yuan, H., Li, G.: A survey of traffic prediction: from spatio-temporal data to intelligent transportation. Data Science and Engineering 6(1), 63–85 (2021)
30. Zhang, M., Liu, Y., Luan, H., Sun, M.: Earth mover's distance minimization for unsupervised bilingual lexicon induction. In: Proceedings of EMNLP (2017)
31. Zheng, K., Zheng, Y., Xie, X., Zhou, X.: Reducing uncertainty of low-sampling-rate trajectories. In: Proceedings of ICDE (2012)
32. Zheng, S., Yue, Y., Hobbs, J.: Generating long-term trajectories using deep hierarchical networks. In: Proceedings of NeurIPS (2016)
33. Zheng, Y.: Trajectory data mining: an overview. ACM Transactions on Intelligent Systems and Technology 6(3), 1–41 (2015)
34. Zhou, F., Gao, Q., Trajcevski, G., Zhang, K., Zhong, T., Zhang, F.: Trajectory-user linking via variational autoencoder. In: Proceedings of IJCAI (2018)