Adversarial Human Trajectory Learning for Trip Recommendation

Qiang Gao, Fan Zhou, *Member, IEEE*, Kunpeng Zhang, Fengli Zhang, and Goce Trajcevski, *Member, IEEE*

***Abstract*— The problem of trip recommendation has been extensively studied in recent years, by both researchers and practitioners. However, one of its key aspects—understanding human mobility—remains under-explored. Many of the proposed methods for trip modeling rely on empirical analysis of attributes associated with historical points-of-interest (POIs) and routes generated by tourists while attempting to also intertwine personal preferences—such as contextual topics, geospatial, and temporal aspects. However, the implicit transitional preferences and semantic sequential relationships among various POIs, along with the constraints implied by the starting point and destination of a particular trip, have not been fully exploited. Inspired by the recent advances in generative neural networks, in this work we propose DeepTrip—an end-to-end method for better understanding of the underlying human mobility and improved modeling of the POIs' transitional distribution in human moving patterns. DeepTrip consists of: a *trip encoder* (TE) to embed the contextual route into a latent variable with a recurrent neural network (RNN); and a *trip decoder* to reconstruct this route conditioned on an optimized latent space. Simultaneously, we define an *Adversarial Net* composed of a *generator* and *critic*, which generates a representation for a given query and uses a critic to distinguish the trip representation generated from *TE* and query representation obtained from *Adversarial Net*. DeepTrip enables regularizing the latent space and generalizing users' complex check-in preferences. We demonstrate, both theoretically and empirically, the effectiveness and efficiency of the proposed model, and the experimental evaluations show that DeepTrip outperforms the state-of-the-art baselines on various evaluation metrics.**

***Index Terms*— Auto-encoder, deep learning, generative adversarial net, spatial-temporal data, trip recommendation.**

I. Introduction

RECENTLY, the high popularity of Location-based Social Network (LBSN) such as Twitter, Flickr, and Instagram, has enabled a generation of massive check-in data with POIs, along with a large number of short messages and other information related to space/location, time, and other context. This, in turn, provides unprecedented opportunities to improve various applications based on mining and learning human mobility, e.g., POI recommendation [1], trip recommendation [2], trajectory-user linking [3], and so on. In particular, the trip recommendation problem has spurred a lot of research since then. In addition to improving route/itinerary planning, it also helps to recover routes of users (e.g., criminals or terrorists) with (partially) missing/anonymized trajectories.

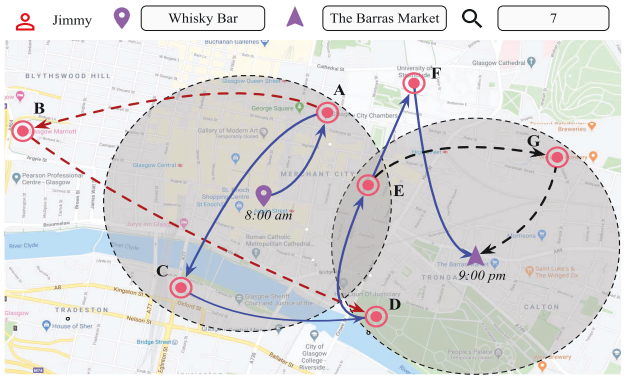


Fig. 1. Illustration example of trip recommendation.

A typical trip recommendation system is to provide a sequence of ordered POIs for a given query which includes, at a minimum, a starting location and a destination. As Fig. 1 shows, Jimmy who lives in the city of Glasgow sends a query request including the start location "*Whisky Bar*" at 8:00 A.M., the destination "*The Barras Market*" at 9:00 P.M., and planning to visit seven sights during that period. The system recommends Jimmy a trip like "*Whisky Bar* $\rightarrow$ *A* $\rightarrow$ *C* $\rightarrow$ *D* $\rightarrow$ *E* $\rightarrow$ *F* $\rightarrow$ *The Barras Market*." This trip recommendation is different from the traditional POI recommendation that aims at suggesting some interesting POIs without any sequential information [4]. It is also different from a typical personalized route planning problem which is usually to search for a shortest route or a route with a minimum cost [5]. Because in the latter problem we only need to find a route from a road network with starting and ending points provided. In contrast, the sequence of POIs recommended in a trip recommender system is not constrained by the road network, and is, arguably, more difficult to obtain due to larger search space.

Thus, the goal of trip recommendation is to build a proper model to capture the various transition distribution among POI

Manuscript received 26 August 2019; revised 1 August 2020; accepted 2 February 2021. Date of publication 23 February 2021; date of current version 5 April 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62072077 and Grant 61602097, in part by the China Scholarship Council under Grant 201906070095, and in part by the NSF Grants CNS under Grant 1646107. *(Corresponding author: Fan Zhou.)*

Qiang Gao, Fan Zhou, and Fengli Zhang are with the School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: qianggao@std.uestc.edu.cn; fan.zhou@uestc.edu.cn; fzhang@uestc.edu.cn).

Kunpeng Zhang is with the Department of Decision Operations & Information Technologies, University of Maryland, College Park, MD 20742 USA (e-mail: kpzhang@umd.edu).

Goce Trajcevski is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: gocet25@iastate.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3058102.

Digital Object Identifier 10.1109/TNNLS.2021.3058102

pairs or a set of POIs, and predict a POI sequence through such a trained model. Therefore, most existing works rely on developing statistical methods to identify top-k popular POIs from massive trajectories [6], applying Markov-based approaches to learn POI transition matrices from a large number of historical trips [7], or adopting certain retrieval algorithms (e.g., Monte Carlo Tree) to make a personalized trip recommendation [2]. These methods first focus on predicting the next POI based on POI transition matrices or attraction popularity when the current POI is given and then formulate a complete POI sequence for a user. These models, in turn, exhibit a similar flow consisting of: 1) integration of spatial and temporal features into each POI, along with some other attributes (e.g., tourist's interest, category of POI, and POI popularity); 2) construction of a model to understand sequential visiting patterns from massive historical trajectories; and 3) development of rank-based or search-based methods to recommend some (ordered) POIs.

The core motivation for our work is based on the observation that the existing studies on trip modeling/planning exhibit certain limitations: 1) first, a trip recommendation is traditionally formulated as a search or statistical problem considering the associated POI information and users' visiting patterns, which is usually not efficient in generating satisfactory routes from massive trajectory data. For instance, many works focus on computing the local transitional distribution between POI pairs by applying Markov-based models [7], [8], given additional user constraints, e.g., spatial distance, duration time, etc., which is computationally intensive due to the estimation of transition probability between all POI-pairs [9]. Moreover, it cannot yield good recommendation since they ignore the high-ordered relations among POIs; 2) second, it is difficult to exploit and integrate the semantics of sequential information among POIs by simply exploring POI popularity or recommending nearby POIs, even some constraints—e.g., the distance between POIs, POI's category, and tourists' social ties—have been considered. In the trip recommended by the system to Jimmy '*Whisky Bar* $\rightarrow A \rightarrow C \rightarrow D \rightarrow E \rightarrow G \rightarrow$ *The Barras Market*', the POI G may not be the location Jimmy expects to visit, if it never appears in his historical location traces; and 3) third, the diversity of the POI sequences based on historical trajectories cannot be obtained through simple empirical analysis. Consider the route's distance distribution for Edinburgh and Melbourne in the Flickr data set, shown in (see [7]) in Fig. 2(a) and (b). Each point indicates the sum of all POI's distances from the start point and distances from endpoint in a given route. Most of the points correspond to short distances and are distributed close to the red dashed line—implying that most of the POIs in a route are closer to either the start point or the endpoint, and the total distance from the start point is similar to the total distance from endpoint. Moreover, even for the same (start_point, end_point) pair, the starting time of the different trips need not be the same.

Some recent works have relied on deep neural network to capture human mobility and the semantics of sequential information among various POIs [3], [10], [11] and constructed recurrent neural networks (RNN) to exploit the

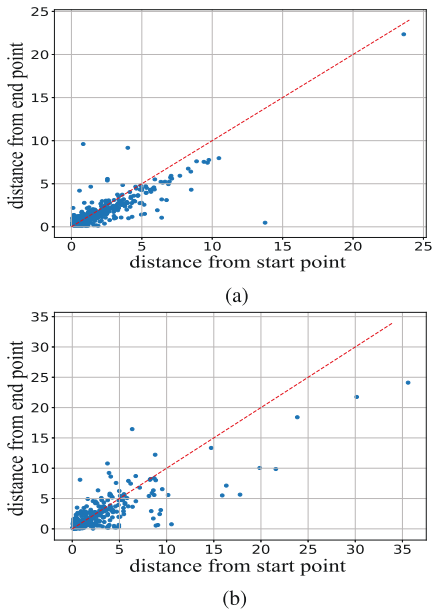


Fig. 2. Route distribution. (a) Distance distribution of Edinburgh. (b) Distance distribution of Melbourne.

POI's transitional distribution (coupled with POI's contextual information, spatial constraints and time preferences. Others have relied on generative models such as adversarial networks [12] and (variational) autoencoders to better exploit the representations of sequential POIs [13], [14]. However, the latent factors governing human's check-in preference are too complex to be captured by an explicit probability density distribution. For instance, Zhou *et al.* [15] use a latent variable model to capture the semantic sequence from human mobility by formulating the latent code sampled from isotropic Gaussian with diagonal covariance. It poses a strong presumption of the trajectory distribution, resulting in ill-defined latent factor inference. Certainly, typical RNN-based models are good at learning the sequential information of a given trip, however, it cannot capture the non-linear transitional patterns among user check-ins. This limits ability to recommend subsequent POIs that do not appear in the training data. Bayesian models such as VAE [16] are appealing on addressing the uncertain transition problem inherent in RNN, but they are also constrained by the agnostic prior distribution problem—i.e., the learned variational distribution of a sequence is usually assumed and regularized to be a diagonal Gaussian, which can lead to posterior collapse and ignoring the latent code—may be exacerbated by combining with RNN-based models [17].

To tackle the aforementioned challenges, we propose *DeepTrip*—an end-to-end neural network in an adversarial spirit, to better understand various context affecting human mobility when making trip recommendations. Specifically, we first introduce a novel POI embedding approach to obtain a low dimensional representation of each POI's contextual information including spatial and temporal constraints. DeepTrip leverages the RNN-based autoencoder as the basis of a framework to study the human mobility patterns via embedding each trajectory/trip into a low dimensional vector. It contains an encoder network and a decoder network, wherein each part is

based on a standard RNN structure such as LSTM [18] and GRU [19]. As such, the sequential and semantic information can be well transformed in a dense vector representation. Meanwhile, the involved RNN can capture the transitional relationships between the elements sourced from sequential data. We then adopt an auxiliary neural network to better learn the sequential POIs distribution in an adversarial manner, so as to sidestep the efforts of explicitly modeling prior distribution with respect to trajectory data.

Preliminary results were reported in [20] and this article advances the prior work in: 1) check-in embedding is a fundamental and significant component in the trip recommendation. Therefore, we propose a novel contextual embedding method which takes the spatial and temporal constraints into consideration. We also provide an explanation regarding why we are integrating these constraints to make performance improved. Extensive experimental results demonstrate the important advantages of such an embedding method; 2) interpretability and theoretical foundations of our proposed methods are detailed in this extended study; and 3) We expand our work via experiments conducted on more data sets (e.g., popular Foursquare data set) and compared with more advanced benchmarks. Overall, our main contributions in this study are fourfold.

1) We take a first step toward addressing the trip recommendation problem under a framework of encoder–decoder, jointly using an adversarial neural network to train a code space for improving the recommendation of preferred routes.
2) We incorporate the geographical and temporal features of each POI, and give special consideration to the distance to the start point and endpoint—in turn, improving the performance of trip modeling in RNN.
3) We encode the route, which contains a series of ordered and discrete POIs, into a latent continuous space that can be used directly in the generative adversarial networks (GANs), avoiding the use of policy gradients or any other auxiliary approaches that traditionally required to tackle discrete data.
4) We present a theoretical interpretation of the proposed model, and conduct extensive experimental evaluations on two real-world data sets, demonstrating the effectiveness and robustness of DeepTrip in comparison to the state-of-the-art baselines.

II. Preliminaries

We now introduce the basic terminology and the background used in the rest of this article.

Let L denote the set of POIs (check-ins left by the users) and for each POI l_τ ($l_\tau \in L$) we have its geographical coordinates $\eth_\tau = < l_\tau^{\text{lat}}, l_\tau^{\text{lon}} >$ and a timestamp t_τ. The formulation of the trip recommendation problem follows the related works [2], [7], [21], [22]:

A. Trip Recommender

A tourist provides a query q that consists the desired start point l_s and start time t_s, the length of trip N (i.e., the number of POIs) and the endpoint l_e at time t_e. A trip recommendation system returns a route $\ell = (l_1, l_2, \ldots, l_N)$, where $l_1 = l_s$ and $l_N = l_e$ for denotation convenience.

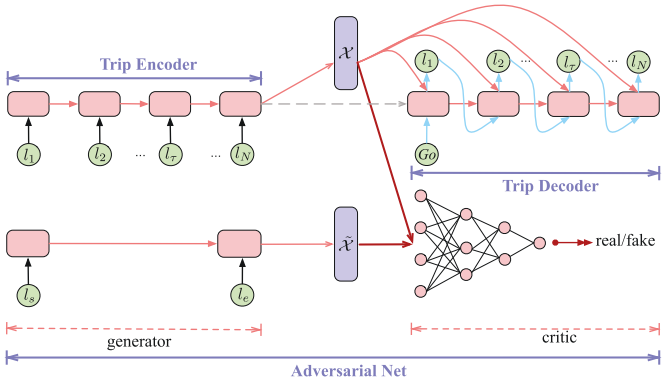


Fig. 3. Overall architecture of DeepTrip.

B. Generative Adversarial Network [23]

is a learning model consisting of a generator $g(\cdot)$ and a discriminator $f(\cdot)$ that compete in a two-player minimax game: $g(\cdot)$ learns to generate fake but reasonable samples, and tries to "fool" $f(\cdot)$; while $f(\cdot)$ attempts to distinguish between the real data, sampled from a true distribution $x \sim \mathbb{P}(\cdot)$, and the samples $\tilde{x}$ that are generated by $g(\cdot)$. The objective of training GAN follows the minimax optimization:

$$\min_{\theta} \max_{\phi} \mathop{\mathbb{E}}_{x\sim\mathbb{P}(x)} [\log f_\phi(x)] + \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g(z)} [\log(1 - f_\phi(g_\theta(z))] \quad (1)$$

where z denotes a latent space obtained by the empirical distribution of the data [e.g., $z \sim N(0, I)$], θ and ϕ are parameterized neural networks of $g(\cdot)$ and $f(\cdot)$, respectively.

C. Wasserstein GAN [24]

It is a novel variant of GAN that replaces the Jensen-Shannon divergence with Earth-Mover distance [a.k.a. Wasserstein-1 distance or optimal transport (OT) distance] [25]–[27], which can tackle the unstable training issue of original GANs. It amounts to optimizing

$$\min_{\theta} \max_{\phi\in\mathcal{W}} \mathop{\mathbb{E}}_{x\sim\mathbb{P}(x)} [f_\phi(x)] - \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g(z)} [(f_\phi(g_\theta(z))] \quad (2)$$

where $\tilde{x} = g_\theta(z)$. In WGAN, $f_\phi(x)$ is alternatively treated as a *critic*, in contrast to the discriminator in GAN. The critic parameters ϕ are restricted to a Lipschitz constraint $\mathcal{W}$ by weight-clipping method, i.e., within a compact space $\phi = [-\epsilon, \epsilon]^d$. For more details on applications of OT distances, including WGAN, We refer readers to the comprehensive surveys in [25] and [26].

III. Methodology

We now present an overview of the proposed DeepTrip model, followed by detailed discussions of each of its modules.

A. Overview of DeepTrip

The overall framework is shown in Fig. 3, and the model is designed with three main components: 1) **Trip Encoder (TE)**: $\mathcal{E}_\omega(\cdot)$; 2) **Trip Decoder (TD)**: $\mathcal{R}_\psi(\cdot)$; and an 3) **Adversarial Net** which further consists of a generator ($g_\theta(\cdot)$) and a critic ($f_\phi(\cdot)$). DeepTrip employs a parameterized *TE* $\mathcal{E}_\omega(\cdot)$ to explore

the spatio-temporal routes, and generates a continuous code space called *trip code* ($\mathcal{X}$). Subsequently, the parameterized *TD* ($\mathcal{R}_\psi(\cdot)$) leverages the *trip code* to recover the input route. The process follows an autoencoder manner, and we incorporate an Adversarial Net to enhance the generation ability of the *TD* by regularizing the autoencoder. We first employ a generator to encode the given query (start point, endpoint and expected length of the trip), to obtain a continuous code representation called *query code* ($\tilde{\mathcal{X}}$), which is similar to the *trip code* $\mathcal{X}$. Then, the critic tries to distinguish the *trip code* $\mathcal{X}$ from the *query code* $\tilde{\mathcal{X}}$ by performing a binary classification. This is a min-max optimization, whereby the generator and critic are trained together with *TE* and *TD*. After the model converges, generator and critic can no longer prevail over each other. Thus, in the recommendation process, we first obtain the *query code* $\tilde{\mathcal{X}}$ by feeding the given input query into the generator of the Adversarial Net, and then input $\tilde{\mathcal{X}}$ into the *TD* $\mathcal{R}_\psi(\cdot)$ to request a recommended route. Clearly, the three components of DeepTrip affect each other: the *trip code* is derived from the *TE* $\mathcal{E}_\omega(\cdot)$, while a recommended route is decoded from the *TD* $\mathcal{R}_\psi(\cdot)$ which relies on the *trip code*, thereby improving the encoded capability of $\mathcal{E}_\omega(\cdot)$. Similarly, the $\mathcal{R}_\psi(\cdot)$ is enhanced by adversarially criticizing the *trip code* and *query code* generated from the given query.

B. Trip Encoder

Recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) [18], [19], have been successfully applied in sequential or time-series data analysis where they take an element one at a time and use a *memory*-like structure to remember information learned from the past. Besides, RNNs recursively encode each element by using the same parameters that can reduce the model complexity and transform independent elements into dependent representations. However, traditional RNNs usually confront the vanishing gradient problem, while LSTM and GRU can alleviate this concern owing to their gated mechanisms. Although GRU is simpler in its structure, LSTM usually achieves more reliable performance than GRU in various real tasks [3], [13]. As such, the *TE* applies an LSTM to capture the POIs' long term dependencies, as well as the spatial and temporal contexts associated with a route. We first obtain the contextual representation of each POI including spatial and temporal constraints and then construct the route as a sequence of POIs, while generating a *trip code* $\mathcal{X}$ via this LSTM.

1) Contextual Embedding: The time and location/distance are critical factors for trip recommendation—see [11], [28], [29] utilize time gate and distance gate to capture spatio-temporal preferences for sequential POIs. However, the existing works focus on integrating the influences of previous spatial locations to the next/current one (as their main task is to make the next POI recommendation). As illustrated in Fig. 2, POIs are often constrained by both of start point and endpoint in term of spatial distribution. Thus, straightforwardly adapting the existing approaches for the next POI recommendation may yield a trajectory that deviates from the expected preferences.

We propose a novel embedding approach, bundling POI's id, spatial and temporal context into a unified dense representation vector that captures the semantics and features better than discrete representations (e.g., one-hot technique) and improves the handling of data sparsity problem. Specifically, we first randomly initiate each POI l_τ with a $d-$dimensional vector $\mathbf{v}(l_\tau) \in \mathbb{R}^d$ by sampling from a truncated Gaussian distribution [28].

Since different users visit a particular POI in different manners, we need to cater to diverse preferences in terms of arrival time. For example, some tourists prefer to visit a museum in the morning, but others may prefer the afternoon. Motivated by previous works on time information processing [28], [30], [31], we construct an hour-level time representation to encode human check-in time preference, which splits the day into 24 discrete time intervals and adopts the continuous variables randomly generated from a truncated Gaussian distribution to represent the 24 hours. Therefore, a tourist who visits a POI at time τ can be represented as $\mathbf{u}(t_\tau) \in \mathbb{R}^{d'}$, where d' is the embedding size.

Then, we represent the geographical feature of POIs with continuous vectors, incorporating the constraints imposed by the start point and endpoint in the representation. They correspond to the distances between current (recommended) POI and the start one, and between current (recommended) POI and the end one, respectively. The geographical feature $\mathbf{u}(\eth_\tau)$ of a POI l_τ is defined as

$$\mathbf{u}(\eth_\tau) = \frac{1}{2}\left(\frac{\text{dist}(\eth_\tau, \eth_s)}{\max_{\text{dist}}}\mathbf{u}(\mathbf{s}) + \frac{\text{dist}(\eth_\tau, \eth_e)}{\max_{\text{dist}}}\mathbf{u}(\mathbf{e})\right) \tag{3}$$

where $\text{dist}(\cdot,\cdot)$ is the great-circle distance, and $\max_{\text{dist}}$ denotes the upper bound of geographic distances in the training data set. $\text{dist}(\eth_\tau, \eth_s)$ denotes the great-circle distance between l_τ and the start point l_s, and $\text{dist}(\eth_\tau, \eth_e)$ denotes the great-circle distance between the l_τ and the endpoint l_e. $\eth_s$ and $\eth_e$ are the geographical coordinates of the start point and the endpoint, respectively. Similar to time embedding, we use two dense vectors, $\mathbf{u}(\mathbf{s})$ and $\mathbf{u}(\mathbf{e})$, to respectively represent the geographical feature of start point and endpoint. In (3), we vectorize the geographical feature of the POI l_τ constrained by the corresponding start point and endpoint in a query. In this way, we formulate the representation of geographical coordinate $\eth_\tau$ of current POI l_τ with the weighted geographical features of the start point and destination.

Finally, we integrate the above three aspects into a unified representation (see Fig. 4) as $\mathcal{C}_\tau = [\mathbf{v}(l_\tau); \mathbf{u}(\eth_\tau); \mathbf{u}(t_\tau)]$, noting that other features associated with POIs (e.g., duration of a visit, if available) can be straightforwardly incorporated in the representation vector.

2) Generating Trip Code: After obtaining the contextual embeddings of POIs, we apply the Long Short Term Memory (LSTM) cells to encode each trip $(l_1, l_2, \ldots, l_N)$ and the current state h_τ^{enc} can be updated by

$$\begin{aligned} i_\tau &= \sigma(W_i\mathcal{C}_\tau + U_i h_{\tau-1}^{\text{enc}} + V_i c_{\tau-1} + b_i) \\ \mathfrak{f}_\tau &= \sigma(W_\mathfrak{f}\mathcal{C}_\tau + U_\mathfrak{f} h_{\tau-1}^{\text{enc}} + V_\mathfrak{f} c_{\tau-1} + b_\mathfrak{f}) \\ o_\tau &= \sigma(W_o\mathcal{C}_\tau + U_o h_{\tau-1}^{\text{enc}} + V_o c_{\tau-1} + b_o) \\ c_\tau &= f_\tau c_{\tau-1} + i_\tau \tanh(W_c\mathcal{C}_\tau + U_c h_{\tau-1}^{\text{enc}} + b_c) \end{aligned} \tag{4}$$

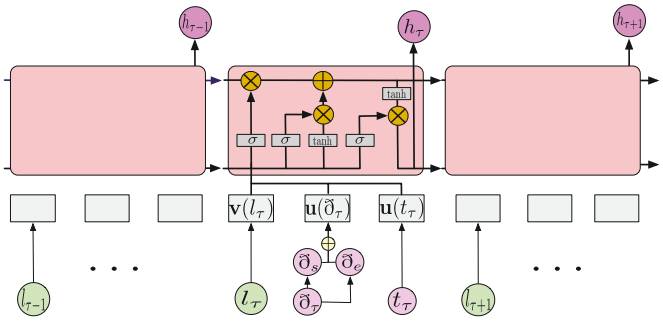


Fig. 4. TE with contextual information.

where i_t, $\mathfrak{f}_t$, o_t, and b_* are respectively the input gate, forget gate, output gate and bias vector; and matrices $W_*, U_*, V_* \in \mathcal{R}^{(d+2d')\times d''}$ are parameters needed to be learned. In (4), each cell state c_τ stores the spatio-temporal information constrained by a given query, which will be transferred to the hidden state h_τ

$$h_\tau^{\text{enc}} = \tanh(c_\tau)o_\tau. \tag{5}$$

Following earlier work [3], the last output of LSTM will be leveraged as the *trip code* $\mathcal{X}$:

$$\mathcal{X} = h_N^{\text{enc}} \tag{6}$$

which can be fed into the *TD* to recover the original route and to have $\mathcal{X}$ updated during the training process.

C. Trip Decoder

The *TD* actually has a dual role of a *decoder* and a *recommender* in the DeepTrip. As a decoder, its objective is to decode the *trip code* $\mathcal{X}$ to reconstruct the input trip. However, unlike the encoder, it generates ordered POIs one by one, predicting the next POI l_τ based on its hidden state h_τ^{dec} and the *trip code* $\mathcal{X}$:

$$l_\tau = W'[h_\tau^{\text{dec}}; \mathcal{X}] + b' \tag{7}$$

where W' and b' denote the weight matrix and bias, respectively. The overall probability of a route $(l_1, l_2, \ldots, l_N)$ can be computed as

$$p((l_1, l_2, \ldots, l_N)) = \prod_{\tau=1}^{N} p\left(l_\tau | h_{\tau-1}^{\text{dec}}, \mathcal{X}\right). \tag{8}$$

The *TE* and the *TD* can be considered as two parameterized functions: the *TE*—$\mathcal{E}_\omega(\cdot)$ generates the *trip code* $\mathcal{X}$ by $\mathcal{E}_\omega(\ell) \mapsto \mathcal{X}$ with parameters ω; and the *TD*— $\mathcal{R}_\psi$ attempts to reconstruct the ℓ from given $\mathcal{X}$ parameterized by ψ. Hence, the reconstruction loss is computed as

$$\mathcal{L}_{ER}(\omega, \psi) = -\log p_\psi(\ell | \mathcal{E}_\omega(\ell)). \tag{9}$$

In order to minimize (9), the cross entropy can be involved to reconstruct the original route ℓ, while the parameters ω and ψ will be updated in the training process [32].

D. Adversarial Net

We now describe the Adversarial Net which we involve to jointly train the autoencoder neural network.

1) Components of Adversarial Net: The two components, as shown in Fig. 3, are the *generator* and the *critic*. A *query code* is obtained from the generator which is constrained by a given query. In turn, the critic will be updated by distinguishing between the *trip code* from *TE* and the *query code* from the generator, whereas the *TE* will be updated by critic and *TD*, simultaneously. This will help to provide an optimal *trip code* space. The generator will also be updated, to provide a better *query code* which would approximate the *trip code* in an adversarial manner. We employ the Wasserstein GAN (WGAN) [33] to construct our Adversarial Net. More specifically, the components can be described as follows.

1) *Generator*: The generator g_θ produces a *query code* $\tilde{\mathcal{X}}$ given a user query $< l_s, N, l_e >$ which is modeled by another LSTM network. Unlike the *TE* and *TD*, it feeds the start-end pair $< l_s, l_e >$ into LSTM cells to generate $\tilde{\mathcal{X}}$ which has the same dimension as the *trip code*

$$\tilde{\mathcal{X}} = g_\theta(\langle l_s, l_e \rangle). \tag{10}$$

2) *Critic*: The critic distinguishes the *query code* $\tilde{\mathcal{X}}$ from the *trip code* $\mathcal{X}$ via performing binary classification. We use a multi-layer fully-connected network as a critic function f_ϕ and, in particular, the critic parameters ϕ are restricted to an 1-Lipschitz function set. Thus, we set each parameter $\phi_i = [-\epsilon, \epsilon]$ $(\phi_i \in \phi)$ following previous works [32], [34].

2) Objective in Adversarial Net: The objective in our Adversarial Net is similar to WGAN [see (2)]—i.e., it follows the min-max optimization in terms of training. It optimizes the parameters θ in the generator and the parameters ϕ in critic, where we show in the following:

$$\min_\theta \max_\phi \mathbb{E}_{\mathcal{X}\sim\mathbb{P}_r}[f_\phi(\mathcal{X})] - \mathbb{E}_{\tilde{\mathcal{X}}\sim\mathbb{P}_f}[f_\phi(\tilde{\mathcal{X}})] \tag{11}$$

where $\mathbb{P}_r$ and $\mathbb{P}_f$ are *trip code* distribution and *query code* distribution respectively. For the training process of our Adversarial Net, we can separate the objective of generator and critic into two parts. For the generator, we define the objective as

$$\mathcal{L}_{\text{Gen}} = \min_\theta \mathop{\mathbb{E}}_{\tilde{\mathcal{X}}\sim\mathbb{P}_f}[f_\phi(\tilde{\mathcal{X}})]. \tag{12}$$

For the critic, we define the objective as

$$\mathcal{L}_{\text{Cri}} = \min_{\phi\in\mathcal{W}} \max_\omega \mathop{\mathbb{E}}_{\ell\sim\mathbb{P}_\ell}[f_\phi(\mathcal{E}_\omega(\ell))] - \mathop{\mathbb{E}}_{\tilde{\mathcal{X}}\sim\mathbb{P}_f}[f_\phi(\tilde{\mathcal{X}})]. \tag{13}$$

Consequently, the Adversarial Net of the DeepTrip will jointly minimize the objectives of (12) and (13).

IV. Algorithmic Aspects

We now provide a detailed analysis of the methodology proposed in Sec. III and present the overall recommendation algorithm.

A. Theoretical Analysis

Generally, when modeling the trip recommendation problem, one can simplify it as fitting an appropriate function that can map the given query q to a requested route ℓ (i.e., $\mathcal{E}_\omega(q) \mapsto \ell$), which can be done by minimizing the following objective:

$$\mathcal{L}'_{ER}(\omega, \psi) = -\sum \log p_\psi(\ell | \mathcal{E}_\omega(q)). \tag{14}$$

Here, $\mathcal{E}_\omega(q)$ actually is a *TE* that will embed the discrete query into a continuous variable. Trip recommendation can also be considered as a problem of constructing a model with latent variable to retrieve a satisfactory route: $\mathcal{E}_\omega(q) \sim \mathcal{X}$. Thus, we define a model to elaborate the training process before we want to make recommendation.

Let $\mathbb{P}_r(\ell)$ be a set of discrete and real (i.e., actual) routes distribution, and $\mathbb{P}(\ell)$ be the model distribution. According to the (14), the training process of the trip recommendation problem is transferred to adapt a conditional distribution $\mathbb{P}(\ell|q)$ by

$$\mathbb{P}(\ell|q) \sim \mathbb{P}(\ell, q)/\mathbb{P}(q). \tag{15}$$

Since any given POI, as well as a collection/sequence of POIs are discrete data items, we attempt to set a continuous latent variable to represent all the discrete data (including the query q and the route ℓ), in which we can preserve the transitional information. Consequently, the two key components to be addressed are:

1) How to force the latent variable of a given query to approximate or match the latent variable of a real route?

2) How to make the latent variable of real route preserve the original information of a route?

We note that this resembles the learning process of autoencoder or variational autoencoder—the autoencoder can be any model trained to force an input data to a latent space. Then we ought to build back the original information [29], [35], which can be done by optimizing the evidence lower bound (ELBO)

$$\log \mathbb{P}(\ell) \geq -KL(\mathbb{Q}(\mathcal{X}|\ell), \mathbb{P}(\mathcal{X})) + \mathbb{E}_{\mathbb{Q}(\mathcal{X}|\ell)}[\log \mathbb{P}(\ell|\mathcal{X})]. \tag{16}$$

Analogously, we create a generator with random parameter θ to implicitly parameterize the unknown real distribution $\mathbb{P}(q)$, and obtain the *query code* $\tilde{\mathcal{X}}$ (see Section III). Specifically, $q \sim \mathbb{P}(q)$, $\tilde{\mathcal{X}} = g_\theta(q)$. Let $\mathbb{P}(\tilde{\mathcal{X}})$ denote the *query code* distribution. We also define the aggregated posterior distribution of the route[1] $\mathbb{Q}(\mathcal{X})$

$$\mathbb{Q}(\mathcal{X}) = \int_\ell \mathbb{Q}(\mathcal{X}|\ell)\mathbb{P}_r(\ell)d\ell. \tag{17}$$

Following [36], we assume that $\mathbb{Q}(\mathcal{X}|\ell)$ is a deterministic function of ℓ. Therefore, $\mathbb{Q}(\mathcal{X})$ is only dependent on the real data distribution $\mathbb{P}_r(\ell)$. Now the main task becomes how to let *query code* imitate or match the *trip code* which is from real route distribution. To accomplish this, we involve the adversarial network to guide the *trip code* distribution $\mathbb{Q}(\mathcal{X})$ toward matching the *query code* distribution $\mathbb{P}(\tilde{\mathcal{X}})$. In essence, this amounts to increasing the similarity between $\mathbb{Q}(\mathcal{X})$ and $\mathbb{P}(\tilde{\mathcal{X}})$, which we evaluate by using the Jensen-Shannon divergence

$$D_G(\mathbb{Q}(\mathcal{X}), \mathbb{P}(\tilde{\mathcal{X}})) = KL(\mathbb{Q}(\mathcal{X})||\mathbb{P}_m) + KL(\mathbb{P}(\tilde{\mathcal{X}})||\mathbb{P}_m) \tag{18}$$

where $\mathbb{P}_m$ is the mixture of $\mathbb{Q}(\mathcal{X})$ and $\mathbb{P}(\tilde{\mathcal{X}})$ [33].

In this article, we adopt WGAN to perform the $D_G(\mathbb{Q}(\mathcal{X}), \mathbb{P}(\tilde{\mathcal{X}}))$ by weight clipping [29], [33].

$$D_G(\mathbb{Q}(\mathcal{X}), \mathbb{P}(\tilde{\mathcal{X}})) = \inf_{\Gamma \in \mathbb{P}(\mathcal{X} \sim \mathbb{Q}(\mathcal{X}), \tilde{\mathcal{X}} \sim \mathbb{P}(\tilde{\mathcal{X}}))} \mathbb{E}_{\mathcal{X}, \tilde{\mathcal{X}} \sim \Gamma}(||\mathcal{X} - \tilde{\mathcal{X}}||) \tag{19}$$

[1]Actually, we assume that $\mathbb{P}(q)$ and $\mathbb{P}(\ell)$ are identical distribution.

where $\mathbb{P}(\mathcal{X} \sim \mathbb{Q}(\mathcal{X}), \tilde{\mathcal{X}} \sim \mathbb{P}(\tilde{\mathcal{X}}))$ is the joint distribution with marginals $\mathbb{Q}(\mathcal{X})$ and $\mathbb{P}(\tilde{\mathcal{X}})$, and '$||\cdot||$' is the Wasserstein distance.

Similar to variational autoencoder [16], [35], a decoder (e.g., a generative model $\mathbb{P}(\ell|\mathcal{X})$), will deterministically reconstruct the route ℓ by mapping *trip code* $\mathcal{X}$ to $\ell = \mathcal{R}_\psi(\mathcal{X})$ which guarantees the quality of the generated latent variable. In other words, we aim to "preserve" the whole information of original data, toward which we are maximizing the expectation by $\mathbb{E}_{\mathbb{Q}(\mathcal{X}|\ell)}[\log \mathbb{P}(\ell|\mathcal{X})]$. Now we can solve out the two key components [1) and 2)] with latent variables, and we need to optimize

$$\underbrace{D_G(\mathbb{Q}(\mathcal{X}), \mathbb{P}(\tilde{\mathcal{X}}))}_{\textbf{(i)}} - \underbrace{\mathbb{E}_{\mathbb{Q}(\mathcal{X}|\ell)}[\log \mathbb{P}(\ell|\mathcal{X})]}_{\textbf{(ii)}}. \tag{20}$$

As Fig. 3 illustrates, in (20), if the Wassertein distance $D_G(\mathbb{Q}(\mathcal{X}), \mathbb{P}(\tilde{\mathcal{X}}))$ between the (distributions of the) *trip code* and the *query code* converges, the generator which encodes the given query can store the latent information about missing POIs in a whole route, because the *query code* distribution is close-enough to the *trip code* distribution. Note that, since the second term converges, it means that the decoder is able to rebuild the route conditioned on a given latent code—moreover, the input latent code can well exploit the sequential information among a set of POIs. Hence, when the (20) converges, we can use the generator to encode the tourist's query and use the obtained latent code to build a route for him/her.

B. Training Algorithm

The objective of the training process of DeepTrip is threefold, according to (9), (12), and (13). First, we need to minimize the reconstruction loss between *TE* and *TD*. Next, the adversarial training on *trip code* and *query code* needs to be employed. Therefore, the total objective of our DeepTrip is to optimize

$$\mathcal{L} = \mathcal{L}_{\text{ER}} + \mathcal{L}_{\text{Cri}} + \mathcal{L}_{\text{Gen}}. \tag{21}$$

1) Computational Complexity: Using (21) above, we can conclude that the computational complexity in $\mathcal{L}_{\text{ER}}$ is dependent on the parameters ω and ψ in each iteration. Hence, it is $\mathcal{O}(n_{\text{iters}} \times (\|\omega\| + \|\psi\|))$, where n_{iters} is the number of iterations. Similarly, the computational complexity of the Adversarial Net, including generator and critic ($\mathcal{L}_{\text{Cri}}$ and $\mathcal{L}_{\text{Gen}}$), is typically related with the parameters θ and ϕ. Therefore, the computational complexity in $\mathcal{L}_{\text{Cri}}$ and $\mathcal{L}_{\text{Gen}}$ is $\mathcal{O}(n_{\text{iters}} \times (k \times \|\phi\| + \|\theta\|))$, where k denotes the iterations used in optimizing the $\mathcal{L}_{\text{Cri}}$ in each iteration of $\mathcal{L}$. Hence, the computational complexity of $\mathcal{L}$ is $\mathcal{O}(n_{\text{iters}} \times (k \times \|\phi\| + \|\theta\| + \|\omega\| + \|\psi\|))$.

The training process of DeepTrip is summarized in pseudo-code in the Algorithm 1. When it comes to the recommendation process by our trained DeepTrip, as illustrated in Fig. 5, we use the trained generator and the trained *TD* to make a trip recommendation for a given query-requested. In the process, we set the *TD* as a recommender to generate the trip for the user. As mentioned, we first aim at obtaining the space code (in the training process, it is called *query code*) from the generator through feeding a testing query. Next, we

Algorithm 1 DeepTrip Training

Input: maximum training epoch n_{iters}, the number of critic training per generator iteration k

/* Training */

initialize Model M

for $epoch = 0$; $epoch < n_{iters}$ **do**

  Minimizing the reconstruction loss between $\mathcal{E}_\omega$ and $\mathcal{R}_\psi$

  Sample $\{\ell_{(i)}\}_{i=1}^{B}$ a batch from route data set

  Compute latent representation $\mathcal{E}_\omega(\ell_{(i)})$

  Obtain the *trip code* $\mathcal{X}_{(i)}$

  Compute $\mathcal{L}_{ER}(\omega, \psi) = -\frac{1}{B}\sum_{i=1}^{B} \log p_\psi(\ell_{(i)}|\mathcal{X}_{(i)})$, backpropagate gradients, update ψ and ω

  Critic training

  for k *steps* **do**

    Sample $\{\ell_{(i)}\}_{i=1}^{B}$ a batch from route data set

    Obtain the *trip code* $\mathcal{X}_{(i)}$

    Compute the adversarial loss on the *trip code* samples $\frac{1}{B}\sum_{i=1}^{B} \mathbb{E}_{l' \sim \mathbb{P}_\ell}[f_\phi(\mathcal{X}_{(i)})]$, backpropagate gradients, update ϕ and ω

    Obtain a batch of *query code* from given query$\{\tilde{\mathcal{X}}_{(i)}\}_{i=1}^{B} = g_\theta(< l_s, l_e >)$

    Compute the adversarial loss $-\frac{1}{B}\sum_{i=1}^{B} \mathbb{E}_{\tilde{\mathcal{X}} \sim \mathbb{P}_g}[f_\phi(\tilde{\mathcal{X}}_{(i)})]$, backpropagate gradients, and update ϕ, clip the weight of the ϕ to $[-\epsilon, \epsilon]^d$

  end

  Generator training

  Obtain a batch of *query code* from given query$\{\tilde{\mathcal{X}}_{(i)}\}_{i=1}^{B} = g_\theta(< l_s, l_e >)$

  Compute the generator loss $\frac{1}{B}\sum_{i=1}^{B} \mathbb{E}_{\tilde{\mathcal{X}} \sim \mathbf{P}_g}[f_\phi(\tilde{\mathcal{X}}_{(i)})]$, backpropagate gradient, and update θ

end

Output: the trained Model M

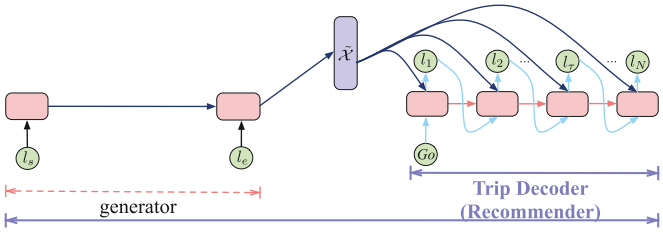


Fig. 5. Recommendation process with DeepTrip.

turn to feed this space code to *TD* while giving the length of recommended POI sequence. More specifically, the route is generated by the *TD*, in which we concatenate the *query code* and the output of each step to obtain the POIs.

V. Experiments

We now present the evaluation of the performance of DeepTrip in comparison with the state-of-the-art trip recommendation methods.[2] Specifically, we address the following questions.

[2]The source code of the implementations is publicly available at https://github.com/gcooq/DeepTrip

TABLE I

Statistics of the Two Data Sets

DataSet	Φ(Visits)	Φ(Trajectory)	Φ(User)
Flickr@Edinburgh	33,944	5,028	1,454
Flickr@Glasgow	11,434	2,227	601
Flickr@Melbourne	23,995	5,106	1,000
Flickr@Osaka	7,747	1,115	450
Flickr@Toronto	39,419	6,057	1,395
Foursquare@Tokyo	8,721	6,414	200

1) *RQ1*. What is the *effectiveness* of DeepTrip—i.e., does it provide better trip recommendation performance?
2) *RQ2*. How important is the impact of the two components - Adversarial Net and Contextual Embedding - in our DeepTrip model?
3) *RQ3*. How good are the recommended results from DeepTrip compared to the existing approaches/baselines? How does the Adversarial Net contribute to the performance of trip recommendation?
4) *RQ4*. How do the parameter settings affect the performance?

The data sets used in our experiments are shown in Table I. The trips in Toronto, Osaka, Glasgow and Edinburgh are extracted from Flickr photos and videos used in [37], while the Melbourne data is as built in [7]. Foursquare data set [38] contains check-ins in Tokyo collected for about 10 months (from April 12, 2012 to February 16, 2013). Each check-in is associated with a timestamp, GPS coordinates and some semantics (e.g., fine-grained venue-categories). We first filter out short trajectories which contain less than three POIs. Furthermore, we normalize the timestamp into hour-level (i.e., mapping each timestamp into 24 intervals). For each city, we first choose trajectories with more than three check-ins, to form the training data sets. Following [7], we adopt leave-one-out cross validation to evaluate all methods.

— *Baselines*: We compare DeepTrip to the following methods:

- POIPopularity [39]: Recommends the most popular or frequently visited POIs to the user at each time instant.
- PersTour and PersTour-L [37]: The trip recommendation is modeled using a formulation of the orienteering problem and considers user trip constraints such as time limits and the need to start and end at specific POIs. PersTour recommends a trip based on exploiting POIs' features with a time budget. And PersTour-L a variant of PersTour by replacing the time budget with a constrained trajectory length.
- POIRank [7]: Recommends a trajectory by first ranking POIs with the RankSVM method, and then connecting them according to their ranking scores.
- Markov and Markov-Rank [7]: Constructs a POI transition matrix, and recommends a trajectory based on Markov transitions and POI ranking.
- Path and Path-Rank [7]: Path and Path-Rank are significant methods for eliminating sub-tours in Markov and Markov-Rank methods by finding the best path using an Integer Linear Program with sub-tour elimination constraints adapted from the Traveling Salesman Problem.

TABLE II
PARAMETER SETTINGS IN OUR EXPERIMENTS

Parameter Settings	Values	Model Settings	Chooses
batch size	8	embedding size	512
learning rate	0.6	hidden size	512
dropout rate	0.5	$\|\mathbf{code}\|$ size	512
critic learning rate	4e-4	ϵ	1.0

• PersQueue [2]: A personalized tour recommendation method that considers the attraction popularity, user interests, and queuing time to make tour recommendation based on Monte Carlo Tree Search (MCTS). Since we do not attempt to constrain explicitly the queuing time of POIs, we employ the time interval between two adjacent POIs as the implicit queuing time for the next POI.

We implemented our model on the TensorFlow platform accelerating by a 1080Ti GPU. For baselines, we used the corresponding source codes[3] provided by [7], and implemented PersQueue based on [2] with implicit duration time feedback.

— *Metrics*: We choose two commonly used metrics for performance comparison, F_1 and pairs-F_1 scores.

$\mathbf{F_1}$ score: We follow [7], [37] in using the F_1 score to evaluate a recommended trip—which is, the harmonic mean of Precision and Recall of POIs in a trip

$$\mathrm{F_1} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{22}$$

pairs-**$\mathbf{F_1}$** score: a specific metric proposed in [7]. It considers both POI correctness and sequential order by measuring the F_1 score of every pair of POIs, whether they are adjacent or not in a trajectory

$$\text{pairs-}\mathrm{F_1} = \frac{2 \times \text{pairs-P} \times \text{pairs-R}}{\text{pairs-P} + \text{pairs-R}} \tag{23}$$

where pairs-P and pairs-R denotes the Precision and Recall of ordered POI pairs respectively. The values of both pairs-F_1 and F_1 are between 0 and 1. The higher the value, the better the recommended results—e.g., a value of 1 means that both POIs and their visiting order in the planned trajectory are exactly the same as the ground truth. We note that the optimal parameter settings tuned for our experiments are shown in Table II (discussed further in Section V-D)

A. Overall Performance (RQ1)

Table III and Table IV show the performance comparisons between the proposed DeepTrip and the baselines, in terms of F_1 and pairs-F_1 scores, respectively. The best results are indicated in boldface font.

First, we observe that DeepTrip outperforms the baselines/state-of-the-art on all the six data sets, with an average improvements 8.72% and 46.52% over the best baseline method in terms of F_1 and pairs-F_1 score, respectively. Among the baselines, the traditional Rank-based methods focus on analyzing various POI's and query's features including geographical and temporal information. Markov-based methods, in contrast, are more concerned with the transitional patterns among POI pairs, and therefore do not perform as well in comparison with Rank-based methods [7]. However, both of these approaches rely on statistical policies that affect the execution time (see Fig. 6), either due to counting the transition distribution or attempting to leverage POIs' information as much as possible. The values in Table III and Table IV demonstrate that it is effective to use the encoder-decoder framework jointly combining the adversarial net in the trip recommendation. One of the main aspects is that spatio-temporal constraints can significantly affect the tourist visiting intentions, and DeepTrip does well in jointly tackling both these aspects by embedding into a deep neural network, providing better performance on a trip recommendation.

B. Effects of Components (RQ2)

Before explicitly addressing **RQ2**, we define four variants of DeeptTrip, each of which focuses on different components. One of the variants is to train tourists' trips by only using the components of TE and TD. We name this method Trip-ED. Specifically, we feed the given query into TE, and leverage TD to make a recommendation. The second variant uses the LSTM cell in DeepTrip without spatial and temporal context, and we call it DeepTrip-L. We derive the third variant DeepTrip-G, which uses the GRU cell instead of LSTM in DeepTrip. At last, we adopt the vanilla GAN [23] instead of the corresponding module in DeepTrip to investigate the impact of adversarial learning, namely DeepTrip-V.

1) Effect of Adversarial Net: Trip-ED which has no Adversarial Net performs poorly compared to DeepTrip as shown by the values in Table V and Table VI. This implies that Trip-ED which uses the encoder-decoder framework cannot exploit well the latent representation from trained routes to formulate the latent representation $\mathcal{X}$. Conversely, this demonstrates that integrating the adversarial network, as done in other variants and DeepTrip, can really improve the performance of the trip recommendation problem.

2) Effect of Contextual Embedding: For our proposed methods with the adversarial network, both DeepTrip-L and DeepTrip adopt the LSTM cell in the TE, TD and the generator net, while DeepTrip-L does not consider the geographical and temporal information. We observe that DeepTrip outperforms DeepTrip-L, which further demonstrates that the geographical constraints and temporal influence affect the tourist's moving preference—which is why we considered each recommended POI's distance, along with the start point and endpoint.

3) Impact of Recurrent Neural Networks: In Table. V and Table. VI, DeepTrip outperforms DeepTrip-G on all three data sets. Despite the similar performance of LSTM and GRU on sequential learning tasks such as machine translation and language generation, we discover that LSTM is more suitable for mobility pattern learning in our task. A similar results can be found in previous human trajectory learning works [3], [40], [41]. We conjecture that this happens due to the sparsity of human mobility data. Thus, the simplified structure in GRU may not be appropriate for sparse sequential data.

4) Impact of GAN Style: From the results, we observe that DeepTrip outperforms DeepTrip-V, which proves the

[3]https://bitbucket.org/d-chen/tour-cikm16

TABLE III

F_1 COMPARISONS AMONG DIFFERENT ALGORITHMS ON FLICKR AND FOURSQUARE DATA

	Edinburgh	*Glasgow*	*Melbourne*	*Osaka*	*Toronto*	*Tokyo*
Popularity	0.701 ± 0.160	0.745 ± 0.166	0.620 ± 0.136	0.663 ± 0.125	0.678 ± 0.121	0.687 ± 0.133
PersTour	0.656 ± 0.223	0.801 ± 0.213	0.483 ± 0.208	0.686 ± 0.231	0.720 ± 0.215	0.701 ± 0.232
PersTour-L	0.651 ± 0.143	0.660 ± 0.102	0.576 ± 0.141	0.686 ± 0.137	0.643 ± 0.113	0.698 ± 0.172
POIRank	0.700 ± 0.155	0.768 ± 0.171	0.637 ± 0.142	0.745 ± 0.173	0.754 ± 0.170	0.728 ± 0.160
Markov	0.645 ± 0.169	0.725 ± 0.167	0.577 ± 0.168	0.697 ± 0.150	0.669 ± 0.151	0.656 ± 0.110
Markov-Rank	0.659 ± 0.174	0.754 ± 0.173	0.613 ± 0.166	0.715 ± 0.164	0.723 ± 0.185	0.677 ± 0.149
Path	0.678 ± 0.149	0.732 ± 0.168	0.595 ± 0.148	0.706 ± 0.150	0.688 ± 0.138	0.663 ± 0.152
Path-Rank	0.697 ± 0.152	0.762 ± 0.167	0.639 ± 0.146	0.732 ± 0.162	0.751 ± 0.170	0.710 ± 0.182
PersQueue	0.470 ± 0.196	0.586 ± 0.231	0.430 ± 0.165	0.507 ± 0.186	0.536 ± 0.187	0.706 ± 0.186
DeepTrip	**0.765 ± 0.148**	**0.831 ± 0.172**	**0.683 ± 0.174**	**0.834 ± 0.166**	**0.808 ± 0.163**	**0.826 ± 0.176**

TABLE IV

PAIRS-F_1 COMPARISONS AMONG DIFFERENT ALGORITHMS ON FLICKR AND FOURSQUARE DATA

	Edinburgh	*Glasgow*	*Melbourne*	*Osaka*	*Toronto*	*Tokyo*
Popularity	0.436 ± 0.259	0.507 ± 0.298	0.316 ± 0.178	0.365 ± 0.190	0.384 ± 0.201	0.255 ± 0.313
PersTour	0.417 ± 0.343	0.643 ± 0.366	0.216 ± 0.265	0.468 ± 0.376	0.504 ± 0.354	0.268 ± 0.157
PersTour-L	0.359 ± 0.207	0.352 ± 0.162	0.266 ± 0.140	0.406 ± 0.238	0.333 ± 0.163	0.261 ± 0.143
POIRank	0.432 ± 0.251	0.548 ± 0.311	0.339 ± 0.203	0.511 ± 0.309	0.518 ± 0.296	0.313 ± 0.377
Markov	0.417 ± 0.248	0.495 ± 0.296	0.288 ± 0.195	0.445 ± 0.266	0.407 ± 0.241	0.210 ± 0.246
Markov-Rank	0.444 ± 0.263	0.545 ± 0.306	0.351 ± 0.220	0.486 ± 0.288	0.512 ± 0.303	0.399 ± 0.233
Path	0.400 ± 0.235	0.485 ± 0.293	0.294 ± 0.187	0.442 ± 0.260	0.405 ± 0.231	0.207 ± 0.195
Path-Rank	0.428 ± 0.245	0.533 ± 0.303	0.344 ± 0.206	0.489 ± 0.287	0.514 ± 0.297	0.271 ± 0.243
PersQueue	0.320 ± 0.243	0.384 ± 0.224	0.335 ± 0.268	0.363 ± 0.283	0.336 ± 0.257	0.232 ± 0.217
DeepTrip	**0.660 ± 0.246**	**0.782 ± 0.257**	**0.546 ± 0.261**	**0.755 ± 0.265**	**0.748 ± 0.247**	**0.643 ± 0.301**

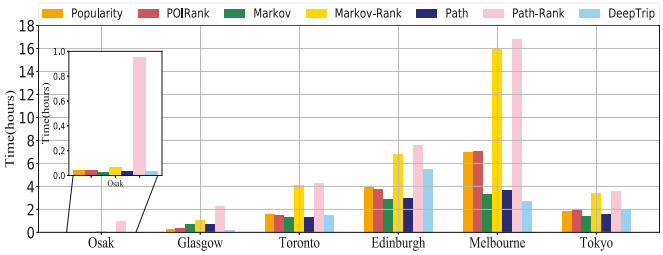


Fig. 6. Runtime comparison on six data sets.

TABLE V

EFFECT OF COMPONENTS ON F_1

	Glasgow	**Melbourne**	**Osaka**
Trip-ED	0.710 ± 0.148	0.605 ± 0.133	0.756 ± 0.163
DeepTrip-L	0.804 ± 0.167	0.681 ± 0.165	0.829 ± 0.165
DeepTrip-G	0.812 ± 0.167	0.671 ± 0.141	0.822 ± 0.169
DeepTrip-V	0.803 ± 0.171	**0.689 ± 0.185**	0.829 ± 0.171
DeepTrip	**0.831 ± 0.172**	0.683 ± 0.174	**0.834 ± 0.166**

effectiveness of WGAN in learning implicit trajectory distribution for better trip recommendation.

C. Interpretability of DeepTrip (RQ3)

In this part, we provide explanations of the recommended results made by DeepTrip. We first visualize the recommended result by a simple example in Edinburgh. There is a user who provides a start location, an end location, and the number of expected visiting POIs (e.g., 5). Fig. 7(a) shows

TABLE VI

EFFECT OF COMPONENTS ON PAIRS-F_1

	Glasgow	**Melbourne**	**Osaka**
Trip-ED	0.659 ± 0.289	0.533 ± 0.214	0.701 ± 0.241
DeepTrip-L	0.738 ± 0.255	0.534 ± 0.255	0.746 ± 0.269
DeepTrip-G	0.750 ± 0.257	0.515 ± 0.252	0.745 ± 0.268
DeepTrip-V	0.724 ± 0.275	0.542 ± 0.277	**0.761 ± 0.265**
DeepTrip	**0.782 ± 0.257**	**0.546 ± 0.261**	0.755 ± 0.265

trip recommendation results by four different approaches, i.e., Markov model, Markov-Rank, Trip-ED, and our DeepTrip based on their respective trained models.

In Fig. 7(a), we discover that the Markov-based method can only successfully make a short POI sequence, because it only considers the last POI to recommend the next POI and is able to capture the local sequential and transitional relationship. However, it also generates some redundancy in the recommended POI sequence [see the yellow star in Fig. 7(a)]. Fig. 7(b) shows a better performance than the Markov-based method—it reduces the redundancies. However, the rank-based method also shows poor performance in capturing the whole sequential distribution. We note that the result of Trip-ED [Fig. 7(c)] is similar to the Markov-Rank method. However, we also note that the proposed DeepTrip, which adopts adversarial nets, gets the best performance and can well capture the global recommended trips' sequential information [Fig. 7(d)].

To further illustrate the performance of adversarial net, we use the t-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize the *trip code* $\mathcal{X}$ and *query code* $\tilde{\mathcal{X}}$

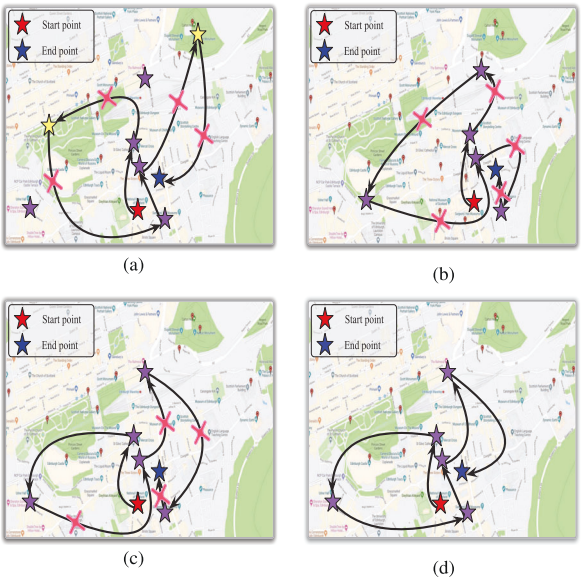


Fig. 7. Recommended results with different methods. (a) Markov. (b) Markov-Rank. (c) Trip-ED. (d) DeepTrip.

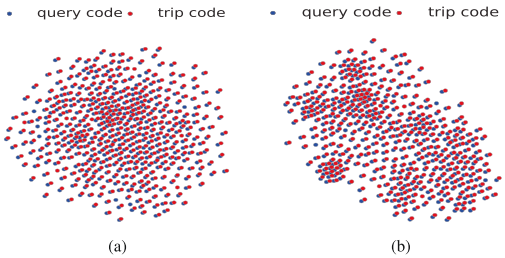


Fig. 8. Visualization of Latent codes. (a) Melbourne. (b) Tokyo.

distribution in 2-dimensional space in Fig. 8. We can see that the *query code* $\tilde{\mathcal{X}}$ distribution is similar to *trip code* $\mathcal{X}$ distribution, which implies that *query code* $\tilde{\mathcal{X}}$ which is only obtained from a given query, indeed affects the generation of the whole sequential information, and can improve the recommendation by feeding the *query code* $\tilde{\mathcal{X}}$ into TD.

D. Effects of Parameters (RQ4)

Lastly, we discuss the influence of several important parameters, including the embedding size of POIs, the hidden size of the neural networks, and the initial learning rate. Besides, the contribution of each training loss will also be discussed.

For embedding size, we present training results with different embedding size of POI in Osaka city and Glasgow city. Fig. 9 demonstrates that a suitable embedding size of POI identity representation can affect the effectiveness of training. We first try to implement our DeepTrip and variants with different embedding size, and aim at selecting the optimal results. In this article, we choose the 256 or 512 as embedding size (see Table II).

For the hidden size, as illustrated in Fig. 10, we also have choices when constructing the training network. In Fig. 11, we show the initial learning rate regarding optimization between the TE and TD in the training process. We set the initial learning rate empirically in our experiments by

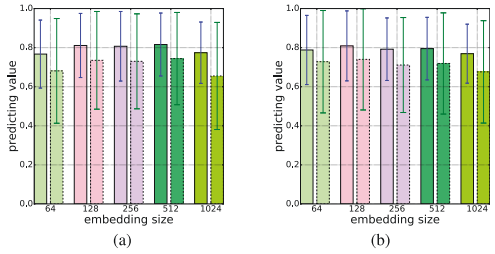


Fig. 9. Results with different embedding sizes. (a) Osaka. (b) Glasgow.

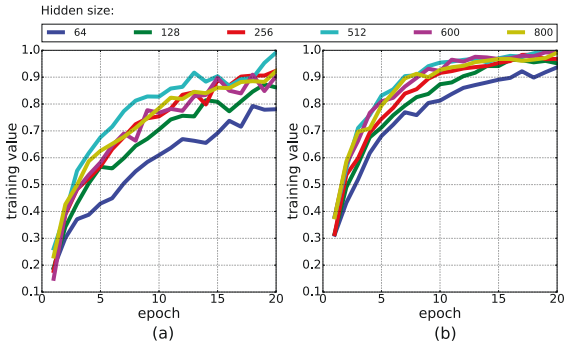


Fig. 10. Training versus hidden size. (a) F_1-score value on Osaka. (b) F_1-score value on Glasgow.

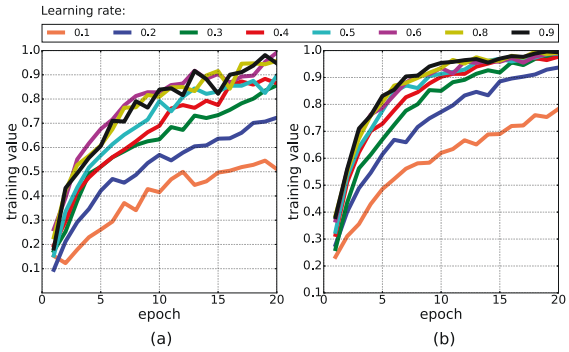


Fig. 11. Training versus learning rate. (a) F_1-score value on Osaka. (b) F_1-score value on Glasgow.

considering the trade-off between effectiveness and over-fitting problem.

Now, we consider whether the reconstruction loss $\mathcal{L}_{\text{ER}}$ and two adversarial losses ($\mathcal{L}_{\text{Cri}}$ and $\mathcal{L}_{\text{Gen}}$) have the same contribution to the model performance of DeepTrip. Toward that, we add a weight (ranging from 0.5 to 2.0) to the two adversarial losses in (21) and the reconstruction loss in (9), respectively. According to the results shown in Fig. 12, we observe that the testing performance is drastically decreased when we gradually decrease the weight of the reconstruction loss. In contrast, the adversarial losses are relatively stable when varying their importance weights. In other words, the reconstruction loss is more sensitive than the adversarial loss.

E. Discussion

Above we have conducted experiments on the data sets of six popular cities to test the effectiveness of our DeepTrip. Compared to the state-of-the-art baselines, our DeepTrip is capable of recommending better trips based on users' specific requests. In addition, we investigate the effect of each key

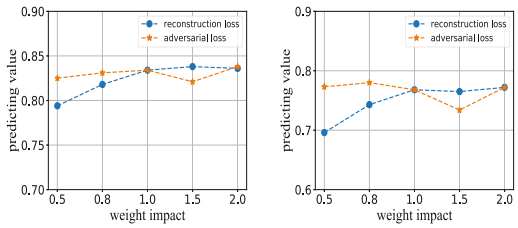


Fig. 12. Results on Osaka with different weight impact. (a) F_1 value. (b) Pairs-F_1 value.

component, the model interpretability, and the effects of some important hyper-parameters to assure the model robustness and generality. Therefore, we can conclude that DeepTrip, which spends more efforts in understanding human mobility through learning implicit trajectory distributions, can provide more pleasant trips while significantly improving model efficiency.

Despite the aforementioned advantages, our model has room for further improvement. First, our model is built on sparse training data, which means it still suffers from the data sparsity issue and the overfitting problem. A possible solution is to augment the mobility data with synthesis but plausible human traces via deep generative models [42]. Besides, the trajectory distribution learned in DeepTrip is inherently intractable. Though we can indeed improve the implicit distribution with adversarial models, the model itself lacks interpretability and explainable and flexible posterior approximation. We hypothesize that tractable density estimation techniques such as normalizing flows [43] may help in interpreting the model behavior.

VI. Related Work

We now review the related literature from three perspectives.

A. POI Recommendation

Most of the existing POI recommendation methods are based on matrix factorization (MF) or Markov Chain (MC) algorithms [4], [8], [44]. There are, however, other approaches leveraging spatial and temporal preference to predict user's next destination or recommend a series of POIs. For instance, a Personalized Ranking Metric Embedding (PRME) method adopts the embedding technique to model user preference [39]. Feng *et al.* leverage a new latent representation model POI2Vec to incorporate the geographical influence for the next POI prediction [1]. Recently, deep learning has been widely used to tackle the POI recommendation problem. A spatial-temporal Recurrent Neural Networks (ST-RNN) was proposed in [11] to model temporal and spatial influences in each time step for POI prediction. A spatio-temporal LSTM model to learn human movement for POI recommendation which more considers POI's distance and time in each LSTM kernel was proposed in [29] and an attentional RNN to predict users' mobility from lengthy and sparse trajectories was proposed in [10]. Recently, a heterogeneous graph-based method [45] and a self-attention-based model [46] have been proposed for improving the POI recommendation accuracy. However, the main task of the POI recommendation problem is to recommend a (ranked) list of top-k POIs to a specific user, which is quite different from a trip recommendation that considers the spatio-temporal constraints and generates a sequence of ordered POIs.

B. Trip Recommendation

The existing solutions for the trip recommendation (some works call this trip planning) are based on variants of the orienteering based problem, where the main idea is to use a heuristic to combine the POIs and trajectories [21]. Wei *et al.* [6] construct top-k popular routes from uncertain trajectory. Zheng *et al.* [47] integrate users' preferences to learn the features of trajectory for personalized trip recommendation. Ge *et al.* [48] consider the time cost in a trip recommendation to learn tourist's interests and the travel cost from traveling tour data. Recently, a novel model called PersTour for recommending personalized tours, which integrated the popularity of POIs and preference of user interest was proposed in [37]. Another proposal for an itinerary recommendation model called TOURMUSTSEE that incorporates the mandatory POIs to search the POI candidate was presented in [22]. A learning model aiming at jointly exploiting the locations' and routes' preferences was proposed in [7], demonstrating that the learning-based methods outperform traditional heuristic trip recommendations. Complementary, the PersQ method incorporates attraction popularity, user interests, and queuing times into consideration for personalized itinerary recommendation [2]. However, these works do not consider the long term dependence of POIs and, arguably, it is difficult to make a diverse route recommendation by simply relying on statistical methods. In addition, they do not consider the semantic information among the POIs in a route.

C. Learning Human Mobility With Generative Model

Deep learning techniques, especially RNNs (e.g., LSTM [18], [49], GRU [19], [50], and bidirectional RNN [51]) have been widely adopted to capture the sequential influence and moving patterns. For example, Gao *et al.* [3] utilizes RNN-based methods to identify human mobility, and Wu *et al.* [52] employs the RNN to model trajectories with topological constraints. Capture both the current and historical human mobility for the next POI prediction by adopting GRU cells was presented in [10].

Deep generative models, such as variational auto-encoder (VAE) [16], [35], [53] and GANs [23], [54], [55], have been widely used in computer vision, natural language processing, and recommendation systems. In particular, deep generative models have successfully been applied in time series modeling by combining Recurrent neural networks [14], [49], [53]. VAE can capture the latent variability from complex high dimensional data and has been successfully used to tackle trajectory classification problem [15] and friendship inference from human mobility [13]. GAN received broad attention due to the ability of generating high-quality image and fluent conversations. They have also been used for human mobility learning, e.g., WGAN [24] has been used to generate synthetic trajectories [12] for the purpose of privacy-preserving of human locations. However, GAN cannot generate discrete data such

as text directly due to lacking the ability of back-propagation through discrete latent variables. Similarly, it is difficult to apply the GAN model directly to exploit human moving patterns. The efforts on tackling the discrete data with the GAN model can be reviewed from two aspects: 1) improving GAN structure itself, e.g., modifying the generator objective by integrating policy gradient algorithm [56], or reparameterizing the categorical distribution with the Gumbel-Softmax trick [57]. These approaches are more concerned about the gradient estimator improvement and need some additional objectives and/or an approximate sampling function; 2) recent popular approach is to firstly transfer or encode the discrete data into a continuous space, and utilize the discriminator to optimize such continuous space, to alleviate the limitations of traditional GANs [24], [32]. In this spirit, our DeepTrip model employs the encoder-decoder framework combining the regularization network [32], toward smoothly learning the latent representation of each trip, thereby improving the trip recommendation.

VII. Conclusion

We presented DeepTrip, a method for learning human mobility based on the adversarial encoder-decoder coupling. We also provided two variants of DeepTrip and conducted extensive experimental evaluations. The results demonstrate superior effectiveness of our proposed approaches in comparison to state-of-the-art benchmarks. To the best of our knowledge, this is among the first work that leverages adversarial networks for learning mobility patterns of trips, by jointly combining the context of each POI and it captures well the human transition patterns for a given start point and endpoint. As part of our future work, we are investigating the way to incorporate other generative models such as variational auto-encoder with Gaussian process [58] and normalizing flows [43], and incorporate more attributes (e.g., check-in category and social preferences) to improve the recommendation performance.

References

[1] S. Feng, G. Cong, B. An, and Y. M. Chee, "Poi2vec: Geographical latent representation for predicting future visitors," in *Proc. AAAI*, 2017, pp. 102–108.

[2] K. H. Lim, J. Chan, S. Karunasekera, and C. Leckie, "Personalized itinerary recommendation with queuing time awareness," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 325–334.

[3] Q. Gao, F. Zhou, K. Zhang, G. Trajcevski, X. Luo, and F. Zhang, "Identifying human mobility via trajectory embeddings," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1689–1695.

[4] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, "GeoMF: Joint geographical modeling and matrix factorization for point-of-interest recommendation," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 831–840.

[5] J. Wang, N. Wu, W. X. Zhao, F. Peng, and X. Lin, "Empowering A* search algorithms with neural networks for personalized route recommendation," in *Proc. KDD*, Jul. 2019, pp. 539–547.

[6] L.-Y. Wei, Y. Zheng, and W.-C. Peng, "Constructing popular routes from uncertain trajectories," in *Proc. KDD*, 2012, pp. 195–203.

[7] D. Chen, C. S. Ong, and L. Xie, "Learning points and routes to recommend trajectories," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 2227–2232.

[8] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," in *Proc. IJCAI*, 2013, pp. 2605–2611.

[9] F. Zhou, H. Wu, G. Trajcevski, A. A. Khokhar, and K. Zhang, "Semi-supervised trajectory understanding with POI attention for end-to-end trip recommendation," *ACM Trans. Spatial Algorithms Syst.*, vol. 6, no. 2, pp. 13:1–13:25, 2020.

[10] J. Feng *et al.*, "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 1459–1468.

[11] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. AAAI*, 2016, pp. 194–200.

[12] K. Ouyang, R. Shokri, D. S. Rosenblum, and W. Yang, "A non-parametric generative model for human trajectories," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Jul. 2018.

[13] Q. Gao, G. Trajcevski, F. Zhou, K. Zhang, T. Zhong, and F. Zhang, "Trajectory-based social circle inference," in *Proc. 26th Int. Conf. Adv. Geographic Inf. Syst. (ACM SIGSPATIAL)*, Nov. 2018, pp. 369–378.

[14] Q. Gao, F. Zhou, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Predicting human mobility via variational attention," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 2750–2756.

[15] F. Zhou, Q. Gao, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Trajectory-user linking via variational AutoEncoder," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3212–3218.

[16] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. NIPS*, 2014, pp. 3581–3589.

[17] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 10–21.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555

[20] Q. Gao, G. Trajcevski, F. Zhou, K. Zhang, T. Zhong, and F. Zhang, "DeepTrip: Adversarially understanding human mobility for trip recommendation," in *Proc. 27th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2019, pp. 444–447.

[21] Z. Friggstad, S. Gollapudi, K. Kollias, T. Sarlos, C. Swamy, and A. Tomkins, "Orienteering algorithms for generating travel itineraries," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 180–188.

[22] K. Taylor, K. H. Lim, and J. Chan, "Travel itinerary recommendations with must-see points-of-interest," in *Proc. Companion Web Conf.*, 2018, pp. 1198–1205.

[23] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.

[24] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, D. Precup and Y. W. Teh, Eds. Sydney, NSW, Australia: PMLR, Aug. 2017, pp. 214–223.

[25] C. Villani, *Optimal Transport: Old and New*, vol. 338. Springer, 2008.

[26] S. Kolouri, S. R. Park, M. Thorpe, D. Slepcev, and G. K. Rohde, "Optimal mass transport: Signal processing and machine-learning applications," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 43–59, Jul. 2017.

[27] J. Yan, X. Liu, L. Shi, C. Li, and H. Zha, "Improving maximum likelihood estimation of temporal point process via discriminative and adversarial learning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2948–2954.

[28] J. Manotumruksa, C. Macdonald, and I. Ounis, "A contextual attention recurrent architecture for context-aware venue recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 555–564.

[29] P. Zhao, H. Zhu, Y. Liu, Z. Li, J. Xu, and V. S. Sheng, "Where to go next: A spatio-temporal LSTM model for next POI recommendation," 2018, *arXiv:1806.06671*. [Online]. Available: http://arxiv.org/abs/1806.06671

[30] T. Qian, B. Liu, Q. V. H. Nguyen, and H. Yin, "Spatiotemporal representation learning for translation-based POI recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, pp. 1–24, Mar. 2019.

[31] B. Hu, M. Jamali, and M. Ester, "Spatio-temporal topic modeling in mobile social media for location recommendation," in *Proc. IEEE 13th Int. Conf. Data Mining*, Dec. 2013, pp. 1073–1078.

[32] J. J. Zhao, Y. Kim, K. Zhang, A. M. Rush, and Y. LeCun, "Adversarially regularized autoencoders," in *Proc. ICML*, 2018, pp. 5897–5906.

[33] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Proc. NIPS*, 2017, pp. 5767–5777.

[34] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," in *Proc. ICLR*, 2018.

[35] D. P Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114
[36] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2015, *arXiv:1511.05644*. [Online]. Available: http://arxiv.org/abs/1511.05644
[37] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized tour recommendation based on user interests and points of interest visit durations," in *Proc. IJCAI*, 2015, pp. 1778–1784.
[38] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 1, pp. 129–142, Jan. 2015.
[39] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new POI recommendation," in *Proc. IJCAI*, 2015, pp. 2069–2075.
[40] C. Miao, J. Wang, H. Yu, W. Zhang, and Y. Qi, "Trajectory-user linking with attentive recurrent network," in *Proc. 19th Int. Conf. Auto. Agents MultiAgent Syst.*, 2020, pp. 878–886.
[41] Q. Gao, F. Zhang, F. Yao, A. Li, L. Mei, and F. Zhou, "Adversarial mobility learning for human trajectory classification," *IEEE Access*, vol. 8, pp. 20563–20576, 2020.
[42] F. Zhou, R. Yin, G. Trajcevski, K. Zhang, J. Wu, and A. Khokhar, "Improving human mobility identification with trajectory augmentation," *GeoInformatica*, vol. 16, no. 7, pp. 1–31, Aug. 2019.
[43] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," 2019, *arXiv:1912.02762*. [Online]. Available: http://arxiv.org/abs/1912.02762
[44] X. Li, G. Cong, X.-L. Li, T.-A.-N. Pham, and S. Krishnaswamy, "Rank-GeoFM: A ranking based geographical factorization method for point of interest recommendation," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2015, pp. 433–442.
[45] M. Hang, I. Pytlarz, and J. Neville, "Exploring student check-in behavior for improved point-of-interest prediction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 321–330.
[46] C. Ma, Y. Zhang, Q. Wang, and X. Liu, "Point-of-interest recommendation: Exploiting self-attentive autoencoders with neighbor-aware influence," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 697–706.
[47] Y. Zheng and X. Xie, "Learning travel recommendations from user-generated GPS traces," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 1, pp. 1–29, Jan. 2011.
[48] Y. Ge, Q. Liu, H. Xiong, A. Tuzhilin, and J. Chen, "Cost-aware travel tour recommendation," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 983–991.
[49] S. Xiao *et al.*, "Learning conditional generative models for temporal point processes," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 6302–6310.
[50] W. Wang *et al.*, "Topic-guided variational auto-encoder for text generation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol. (NAACL-HLT)*, Minneapolis, MN, USA, vol. 1, Jun. 2019, pp. 166–177.
[51] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[52] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3083–3090.
[53] L. Li, J. Yan, X. Yang, and Y. Jin, "Learning interpretable deep state space model for probabilistic time series forecasting," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2901–2908, doi: 10.24963/ijcai.2019/402.
[54] R. Li, L. Li, X. Wu, Y. Zhou, and W. Wang, "Click feedback-aware query recommendation using adversarial examples," in *Proc. World Wide Web Conf. (WWW)*. New York, NY, USA: ACM, 2019, pp. 2978–2984, doi: 10.1145/3308558.3313412.
[55] S. Xiao, M. Farajtabar, X. Ye, J. Yan, L. Song, and H. Zha, "Wasserstein learning of deep generative point process models," in *Proc. Adv. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2017, pp. 3247–3257.
[56] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proc. AAAI*, 2017, pp. 2852–2858.
[57] M. J. Kusner and J. M. Hernández-Lobato, "GANS for sequences of discrete elements with the gumbel-softmax distribution," 2016, *arXiv:1611.04051*. [Online]. Available: http://arxiv.org/abs/1611.04051
[58] Z. Dai, A. Damianou, J. González, and N. Lawrence, "Variational auto-encoded deep Gaussian processes," 2015, *arXiv:1511.06455*. [Online]. Available: http://arxiv.org/abs/1511.06455

Qiang Gao received the B.S. degree in information management and information system from Anhui Polytechnic University (AHPU), Wuhu, China, in 2014, and the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2020.

His current research interests include spatio-temporal data processing via deep learning, location based social network analysis, and human mobility mining.

Fan Zhou (Member, IEEE) received the B.S. degree in computer science from Sichuan University, Chengdu, China, in 2003, and the M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2006 and 2012, respectively.

He is currently an Associate Professor with the School of Information and Software Engineering, University of Electronic Science and Technology of China. His research interests include machine learning, neural networks, spatio-temporal data management, graph learning, recommender systems, and graph data mining and knowledge discovery.

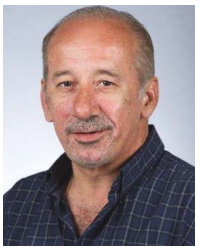
Kunpeng Zhang received the Ph.D. degree in computer science from Northwestern University, Evanston, IL, USA, in 2013.

He is currently a Researcher in large-scale data analysis with particular focuses on mining social media data through machine learning, network analysis, and natural language processing techniques. He is currently an Assistant Professor with the Department of Information Systems, Smith School of Business, University of Maryland, College Park, MA, USA. He has authored or coauthored articles in the area of social media, text mining, network analysis, and information systems on top conference and journals.

Dr. Zhang serves as program committees for many international conferences and is an Associate Editor for *INFORMS Journal on Computing and Electronic Commerce Research Journal*.

Fengli Zhang received the B.S. degree in computer science and the M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1983, 1986, and 2007, respectively.

She is currently a Professor with UESTC. Her research interests include machine learning, network security, and spatio-temporal data management.



Goce Trajcevski (Member, IEEE) received the B.Sc. degree from the University of Sts. Kiril i Metodij, Skopje, North Macedonia, in 1989, and the M.S. and Ph.D. degrees from the University of Illinois at Chicago, Chicago, IL, USA, in 1995 and 2002, respectively.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA. He has coauthored over 140 publications in refereed conferences and journals, in addition to a book chapter and three encyclopedia chapters. His research has been funded by the National Science Foundation, ONR, BEA, and Northrop Grumman Corporation. His main research interests are in the areas of spatio-temporal data management, uncertainty and reactive behavior management in different application settings, and incorporating multiple contexts.

Dr. Trajcevski was the General Co-Chair of the IEEE ICDE 2014, ACM SIGSPATIAL 2019, the PC Co-Chair of the ADBIS 2018 and ACM SIGSPATIAL 2016 and 2017, and has served in various roles in organizing committees in numerous conferences and workshops. He is an Associate Editor of the *ACM Transactions on Spatial Algorithms and Systems and Geoinformatica*.