DeepTrip: A Deep Learning Model for the Individual Next Trip Prediction With Arbitrary Prediction Times

Pengfei Zhang, Haris N. Koutsopoulos, and Zhenliang Ma

***Abstract*— The increasing availability of travel trajectory data allows for a better understanding of travel behavior. In the individual mobility analysis, the problem of next trip prediction assumes a central role and is beneficial for applications such as personalized services and mobility management. This paper addresses the next trip prediction problem with arbitrary prediction times (the time when the prediction is made). This problem has not been studied adequately in the literature and it is important for applications driven by system events, such as proactive travel recommendations under disruptions or crowding in transport systems. It predicts an individual's next trips given their historical trip sequences and the prediction time. We formulate the next trip prediction problem as on-board and off-board predictions depending on an individual's travel status (i.e. on-board/off-board). Using historical/real-time travel trajectories, a DeepTrip model is proposed based on a trip sequence-to-sequence deep learning structure coupled with an attention mechanism. A novel overlapped embedding method is proposed to represent continuous travel attributes capturing simultaneously the categorical and numerical feature information. We also develop a random-sampling training algorithm to learn the impact of the prediction time. The model is validated using trip data in urban rails. The results show that DeepTrip outperforms statistical-based models by more than 10% in terms of accuracy and other deep learning models by 2%-3%. The impact analysis shows that different representations are appropriate for the two prediction cases (on-board/off-board), and the prediction performance does not monotonically improve as the prediction time approaches the next trip.**

***Index Terms*— Next trip prediction, individual mobility, deep learning, metro systems.**

I. Introduction

INDIVIDUAL mobility studies how humans move within a network or system [1]. Understanding and predicting individual mobility is essential and beneficial for many applications in areas such as urban planning [2], [3], personalized recommendations [4], and Intelligent Transportation System (ITS) [5]. Individual mobility prediction can be generally defined as follows: given a series of spatiotemporal records $S = [r_1, r_2, \ldots, r_n]$, referred to as individual mobility records hereafter, predict the next mobility records r_{n+1}. Depending on different prediction tasks, r_i can be a timestamped location in GPS trajectory data (next location prediction), or a trip record with origin/destination times/locations (next trip prediction).

Depending on the application context, the individual mobility prediction can be categorized into two classes: customer event triggered and system event triggered. In the customer event-triggered mobility prediction, r_{n+1} is predicted when mobility record r_n is collected. For example, in personalized recommendations, once the latest mobility record (r_n) of an individual is observed, the next possible visiting location and time ($r_{n+1} = (l_{n+1}, t_{n+1})$) are predicted so that recommendations about the predicted location could be timely pushed [6]. The individual mobility prediction triggered by a system event may take place *at any time* serving a certain system purpose. In this case, the time when the prediction is made (prediction time) is important. Personalized information provision is increasingly becoming an important service of smart transportation systems. With the ability to predict users′ mobility at arbitrary times, transportation authorities could target the affected users in certain situations and provide them with relevant information. For example, the model may predict the potentially affected passengers after an incident takes place in a metro system. Disruption information could then be pushed only to these users. Also, operators could identify potential travelers during the morning peak beforehand and provide them with incentive schemes [7] to reduce system congestion during peak hours. Compared to the traditional information system (e.g., distributing the information on the website or pushing the information to all users), the prediction-based personalized information provision can improve the information relevance and accessibility, as well as reduce the communication cost.

The customer event-triggered mobility prediction is defined as: given a sequence of an individual's historical mobility records $[r_1, r_2, \ldots, r_n]$, predict their next travel record r_{n+1} immediately after r_n is observed. Different from that, the system event triggered mobility prediction is defined as: given a sequence of an individual's historical mobility records $[r_1, r_2, \ldots, r_n]$, predict the next travel record r_{n+1} at time

Manuscript received 18 November 2021; revised 8 June 2022 and 21 December 2022; accepted 20 February 2023. Date of publication 16 March 2023; date of current version 31 May 2023. This work was supported in part by the Kungliga Tekniska högskolan (KTH) Digital Futures Project on cAIMBER through the Basic Research Fund under Grant 230620057 and in part by the Startup Research Fund of Henan Academy of Sciences under Grant 231820013. The Associate Editor for this article was F.-Y. Wang. *(Corresponding author: Zhenliang Ma.)*

Pengfei Zhang is with the Institute of Physics, Henan Academy of Sciences, Zhengzhou 450007, China (e-mail: zhangpfscut@gmail.com).

Haris N. Koutsopoulos is with the Department of Civil and Environmental Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: h.koutsopoulos@northeastern.edu).

Zhenliang Ma is with the Department of Civil and Architectural Engineering, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden (e-mail: zhema@kth.se).

This article has supplementary downloadable material available at https://doi.org/10.1109/TITS.2023.3252043, provided by the authors.

Digital Object Identifier 10.1109/TITS.2023.3252043

$t \geq t_n$. The customer event-triggered mobility prediction problem has been widely studied in areas such as recommendations for points of interest (e.g. dining/tour) [8] and next trips in public transport [9]. However, to the best of our knowledge, the system event triggered individual mobility prediction problem in which the prediction time is arbitrary has not been addressed in the literature. The customer event-triggered mobility prediction problem is a special case of the system event-triggered prediction problem when the prediction time is set as t_n.

The system event-triggered individual mobility prediction problem is challenging in the following aspects:

- It is challenging to model the prediction time capturing these impacts. Figure 1a shows the definition of the individual mobility prediction problem. Existing studies predict the mobility attributes given the individual's travel sequence. As discussed, including arbitrary prediction time in the prediction is essential for online applications. We model the problem as predicting the next trip given both the travel sequence and the prediction time information. Figure 1b is an example of such case. Assume this individual has two travel patterns for work-to-home trips starting at 16:00 and 19:00, respectively. Given that the last trip (trip n) happens at 8 o'clock in the morning, predicting at $t_1 \leq$ 16:00 has to consider that the next trip may take place at 16:00 or 19:00. However, predicting at $t_2 > 16:00$ could utilize the additional information that the trip at 16:00 did not happen. Thus the next trip will probably be the 19:00 trip.
- The mobility sequence has varied types of data associated with it, including discrete (e.g. labeled metro stations) and continuous data (e.g. trip time). The continuous time data in existing studies is discretized into intervals [9], [10]. However, this approach suffers from the interval length and boundary issues. For example, 8:59 AM and 9:01 AM are represented by two different classes if time is discretized into hourly intervals starting on one hour.
- Given information on the travel sequences of an individual, capturing the complex spatiotemporal and historical dependencies, as well as selecting the most relevant sequences (context-aware) for prediction is not straightforward.

The paper proposes a deep learning-based framework, DeepTrip, to model the system event-triggered individual mobility prediction problem that overcomes limitations in the existing literature. The main contributions are:

- Development of a deep natural language processing (NLP) based model structure to predict, at arbitrary prediction times, individuals' next trips given their historical trip sequences. The model can also deal with arbitrary prediction horizons (when the next trip happens) not necessarily daily.
- Development of novel data representation methods for discrete and continuous attributes of trip sequences, using an overlapped embedding model for temporal data representation. The model captures both the categorical and numerical features of temporal information.
- Development of a random-sampling training algorithm coupled with a pairwise time-pointer mechanism to capture the impact of the arbitrary prediction time.
- Empirical analysis to validate the model using smart card trip data from a busy urban railway system and comparing with state-of-the-art models, as well as systematically explore the impacts of data representations and prediction time on model performance.

The remainder of the paper is organized as follows. Section II reviews relevant literature on individual mobility focusing on problem definition, data representation, and prediction models. Section III defines the problem, proposes the DeepTrip model and details the prediction methodology. A case study using smart card data from an urban railway system is performed in section IV. The model performance is validated by comparing it with state-of-the-art models. The impacts of the data representation and prediction time are investigated. Finally, section V concludes the paper and discusses further studies and applications.

II. Literature Review

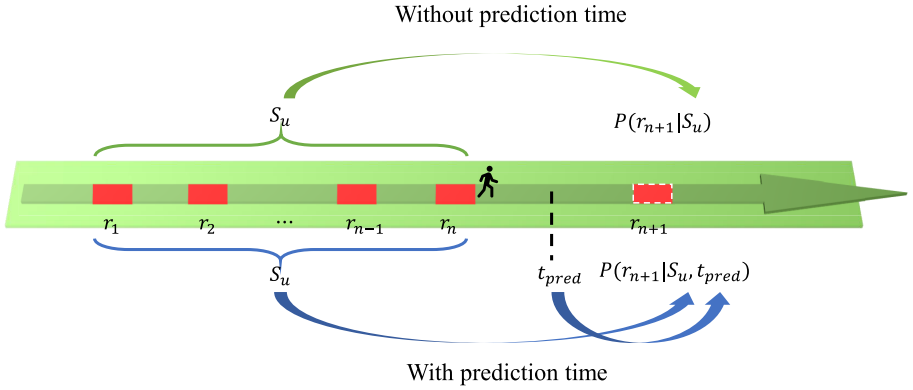
Various studies have shown that individual mobility, in general, can be predicted depending on mobility characteristics [11], [12], [13]. For example, Song et al. [11] used an entropy measurement and showed that 93% of human mobility can be potentially predicted in terms of the next location using mobile call and dial data. The rapid advancements in learning algorithms facilitate urban mobility prediction development. We reviewed the literature from three perspectives: Problem setting, data representation, and prediction models.

A. Problem Setting

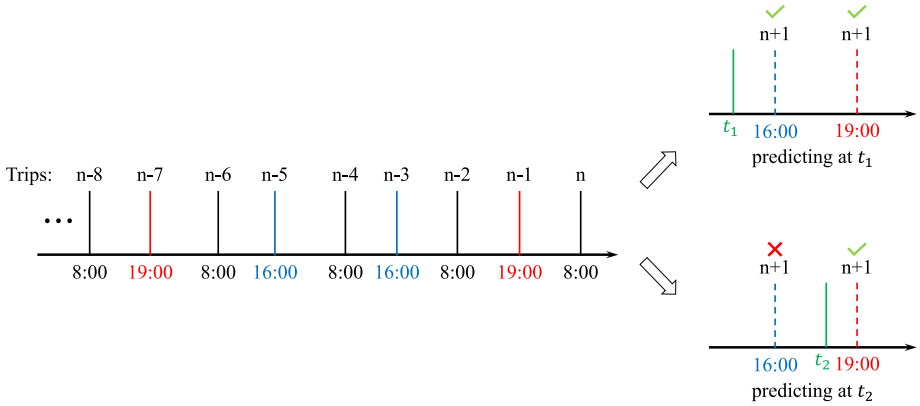
The problem of individual mobility prediction has been studied under different contexts. Early studies focus on the individual next location prediction. For example, Calabrese et al. [6] assume that the human mobility behavior is periodic over time with a period T and predict the location at time t based on the sequence of locations at times $t - T$, $t - 2T$, etc. Many studies apply a Markov Chain model and predict the next location based on transition probabilities of candidate locations [14], [15], [16]. Recently, some studies are reported on predicting both the next locations and times. The prediction is triggered by a customer event, which is predicting the next travel record right after the most recently recorded trip (location and time) are observed. For example, Gidófalvi and Dong [8] propose a continuous time Markov model to predict individual departure times and destinations.

The aforementioned models focus on the next location prediction, however, very limited attention has been paid to the next trip prediction (origin, destination, and times). Zhao et al. [9] developed a mobility n-gram model to predict passengers' next trips in transit systems using AFC data. The approach divides the prediction into two sub-tasks: trip making prediction and trip attributes prediction, modeled using logistic regression and n-gram models, respectively.

Different from these problems, this paper formulates the individual mobility prediction problem with arbitrary prediction times (when the prediction is made) and the prediction



(a) Individual mobility prediction with arbitrary prediction time. Different from the existing prediction problem $P(r_{n+1}|S_u)$ in literature, the paper predicts the mobility attributes given travel sequence and prediction time, i.e. predicting $P(r_{n+1}|S_u, t_{pred})$, where t_{pred} is the prediction time point. How to capture the information interaction between S_u and t_{pred}, and figure out how they influence the predicted probability distribution is challenging.



(b) Illustration of the impact of prediction time on performance. Given the regular travel pattern between 8:00-16:00 and 8:00-19:00, if predicting at t_1, 16:00 and 19:00 are both possible to be the candidate, while 16:00 will be filtered out if predicting at t_2.

Fig. 1. Individual mobility prediction and the importance of prediction time information.

horizon that may go beyond the end of the calendar day. Such a model can be useful for applications driven by system events, such as proactive travel recommendations under disruptions or crowding in transport systems. The model, for a given prediction time, predicts the individual next trips given the individual's historical trip sequences. The problem studied in this paper generalizes existing problems that assume a prediction time defined by the end of the last trip, and a prediction horizon constrained by the end of the day.

B. Data Representation

Mobility records contain the spatial-temporal travel attributes of trips, e.g. origin, destination, departure time, etc. Properly representing these attributes is critical for individual mobility prediction. The spatial travel attributes are usually represented using a grid-based system. For example, Calabrese et al. [6] divided the geographical space into labeled grids and allocated GPS records to the corresponding grids based on their coordinates. Feng et al. [10] aggregate GPS records with the proximity in space and time into one proxy record. Instead of directly using the location information, Gambs et al. [15] mined Point of Interest (POI) data to represent the corresponding travel locations. POI is found to be more informative than location information. They can capture the underlying travel activities and purpose, and thus improve the individual prediction performance.

Representing temporal information is generally challenging. Ideally, the temporal information representation includes both categorical and numerical features. The categorical features refer to the semantic relationship between time points regardless of the magnitude of the time difference, while the numerical features capture the chronological dependencies, particularly for discretization boundaries. Regarding temporal information representation, the simplest form used in the literature is based on the order of visiting different locations, regardless of the difference of these times [14], [15], [16]. For representations incorporating time attributes, two main approaches are commonly used. The first considers only the time of day but ignores the date. For example, Feng et al. [10] divided the day into 24 one-hour intervals. The drawbacks of such representation are: a) the proper interval width is hard to determine; b) the representation is sensitive to interval boundaries (two close time points on the two sides of the interval boundary are treated as very different). Several methods have been proposed to address these issues, including entropy-based and error-based discretization [17]. Although these methods could figure out a more reasonable division of

the time domain, boundary and interval width issues still exist. The second approach uses timestamp records with both date and time information. The timestamp records are transformed into a sequence of ordered integers [8]. The limitation of both approaches is that the numerical feature of time is over-represented, not well representing the temporal heterogeneity of mobility activities. For example, the same mobility activity may start/end with a time difference from day to day. Using the exact timestamp value may cause the model to treat the same activity differently, thus degrading the prediction performance.

We propose a novel overlapped embedding model to capture both categorical and numerical features of time attributes and automatically switch between them given the prediction context. The structure fits well with the nature of the studied mobility prediction problem.

C. Prediction Models

The individual mobility prediction models in the literature can be categorized as statistical and deep learning-based models.

The statistical models have been widely used and they model the mobility sequence dependencies using probabilistic methods, such as the Markov chain and its variations [15], [18], [19], decision trees [20], and natural language models [21]. For example, Gambs et al. [15] developed a n Mobility Markov Chain (n-MMC) model to predict the next location based on n previous visited locations. The n-MMC model exhibits a promising performance when $n = 2$. Increasing the number of looking back intervals has no significant improvement if $k > 2$. Mathew et al. [18] proposed a hybrid approach to predict the next location using GPS data by combining Hidden Markov Models (HMMs) and location clustering. The individual visited locations are clustered and then fed into the HMMs. Zhang et al. [19] developed a group-specific mobility modeling framework and predict the next visiting location based on Geo-tagged social media data. Monreale et al. [20] proposed a T-pattern tree model, a decision tree-based model, which learns from a previously extracted concise representation of mobility behaviors using GPS trajectory data. Hsieh et al. [21] developed a time-aware language model (T-gram) to predict the next location using check-in data.

Despite their performance, the statistical-based prediction models can only capture the transition probability of mobility patterns in a static manner (calibrated or learned from the training data). They are limited in utilizing longitudinal dependencies, such as periodical regularity. Deep learning-based individual mobility prediction methods have emerged in recent years [10], [22], [23]. They not only capture the high-order spatial-temporal dependencies but also learn the longitudinal and periodic features, thus providing a better performance compared to statistical models. For example, Feng et al. [10] proposed the DeepMove model, a sequence-to-sequence model structure coupled with an attention mechanism filtering historical sequences, to predict an individual's next visiting location. The results show that DeepMove outperformed the MMC model by more than 10% in terms of accuracy. Rossi et al. [22] combined a Long-Short-Term-Memory (LSTM) network and an attention module to predict the next destination of taxis. Recently, newly emerged deep sequential models have been incorporated into deep learning-based mobility prediction models. Xue et al. [24] proposed a transformer-based model, MobTCast, to predict the traveler's next visiting location; Tao et al. [25] combined the transformer model with reinforcement learning to predict the individual's mobility for long-term. Other prediction scenarios have also been studied. Zhao et al. [26] studied the casual trip prediction in the metro system, and Zhou et al. [27] focused on the sparse trajectory prediction and trajectory classification.

The DeepTrip model proposed in this paper predicts the individual next trip attributes under the system event-triggered context with a deep learning-based framework.

III. Methodology

A. Problem Definition

Table I summarizes the notations used in the paper.

Let the time domain be denoted by $\mathbb{T}$ and be modeled as the ordered set of non-negative natural numbers $\mathbb{N}^+$. A trip tr of an individual can be characterized by a tuple of the related attributes where $tr = (a_1, a_2, \ldots, a_p)$. a_i denotes the i^{th} attribute. In this study, we use 4 attributes—start time t_o, origin o, destination d, and day of the week w, to formulate the attribute tuple where $tr = (t_o, o, d, w)$. Note that extra attributes could also be added to the attribute tuple if available. Also, we do not include end time t_d of a trip into the tuple since t_d is mainly dominated by the system and not the individual's mobility behavior, predicting t_d is out of the scope of this study.

The trip sequence of an individual u is defined as a chronological list of u's last n recorded trips where $S_{u,n} = [tr_1, tr_2, \ldots, tr_n] = [(t_{o,1}, o_1, d_1, w_1), \ldots, (t_{o,n}, o_n, d_n, w_n)]$, where tr_n denotes the recorded trip at time instance n, i.e. the last observed trip of u. The trip start times are irregularly spaced but temporally ordered where $t_{o,1} < t_{o,2} < \cdots < t_{o,n} \in \mathbb{T}$.

Let $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$ denote the set of the visited stations in the metro system. For an individual, $o_i, d_i \in \mathcal{R}$, $\forall i \in 1, 2, \ldots, n$. Let $\mathcal{W} = \{1, 2, \ldots, 7\}$ denote the set of the days of the week, i.e. from Monday (1) to Sunday (7). For an individual, $w_i \in \mathcal{W}$, $\forall i \in 1, 2, \ldots, n$.

Based on the individual's travel status, the mobility prediction problem is divided into two sub-problems: off-board prediction and on-board prediction. The off-board prediction problem is informally defined as predicting each travel attribute of the following trip.

- *Problem 1. Off-Board Prediction:* Given an individual u, its trip sequence history $S_{u,n}$ up to time instance n, and prediction time t_{pred} where $t_{d,n} < t_{pred} < t_{o,n+1}$, predict the origin o_{n+1}, destination d_{n+1}, start time $t_{o,n+1}$, and day of the week w_{n+1} of the next trip.

Subsequently, the on-board prediction problem is informally defined as predicting the destination of the current trip (the prediction time is between the start time and completion time of a trip).

- *Problem 2. On-Board Prediction:* Given an individual u, its trip sequence history $S_{u,n}$, partial information of trip

TABLE I
NOTATION

Variable	Description
u	An individual
tr_i	Attributes of i^{th} trip, including origin, destination, origin time, and day of the week
o_i	Origin of trip tr_i
d_i	Destination of trip tr_i
$t_{o,i}$	Departure time from origin of trip tr_i
$t_{d,i}$	Arrival time of destination of trip tr_i
t_i^0, t_i^{24}	0:00 AM and 24:00 PM of the day that the trip tr_i happens, respectively
w_i	Day of the week of trip tr_i
S	Trip sequence of an individual, where $S = [tr_1, tr_2, \ldots, tr_n]$
S_h	$S_h = [tr_1, tr_2, \ldots, tr_{k-1}]$, historical trip sequence of an individual with respect to last trip n
S_c	$S_c = [tr_k, tr_{k+1}, \ldots, tr_n]$, real-time trip sequence of an individual with respect to last trip n
t_{pred}	Prediction time, i.e. the time when the prediction is conducted
$\boldsymbol{v}_i$	Feature vector of trip tr_i
S_{V_h}	Historical feature vector sequence, where $S_{V_h} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{k-1}]$
S_{V_c}	Real-time feature vector sequence, where $S_{V_c} = [\boldsymbol{v}_k, \boldsymbol{v}_{k+1}, \ldots, \boldsymbol{v}_n]$
$\boldsymbol{h}_i^h$	Hidden state vector of $\boldsymbol{v}_i$, $\forall \boldsymbol{v}_i \in S_{V_h}$
$\boldsymbol{h}_i^c$	Hidden state vector of $\boldsymbol{v}_i$, $\forall \boldsymbol{v}_i \in S_{V_c}$
$\boldsymbol{c}_n$	Context vector of the last trip n
g_i	Gap time of trip tr_i, it is the time difference between $t_{d,i-1}$ and $t_{o,i}$
tp_g^1	Gap time time-pointer 1 (time difference between trips) in off-board prediction, where $tp_g^1 = [g_1, g_2, \ldots, g_n]$
tp_g^2	Gap time time-pointer 2 (elapsed time from last trip n) in off-board prediction, where $0 \leq tp_g^2 < g_{n+1}$
τ^i	Duration time of trip tr_i, it is the time difference between $t_{d,i}$ and $t_{o,i}$
tp_τ^1	Duration time time-pointer 1 (trip travel time) in on-board prediction, where $tp_\tau^1 = [\tau_1, \tau_2, \ldots, \tau_n]$
tp_τ^2	Duration time time-pointer 2 (elapsed time from last boarding) in on-board prediction, where $0 \leq tp_\tau^2 < \tau_{n+1}$
$\mathcal{L}_{pred}$	Loss function of the next trip prediction task
$\mathcal{L}_p$	Penalty term of multi-head attention

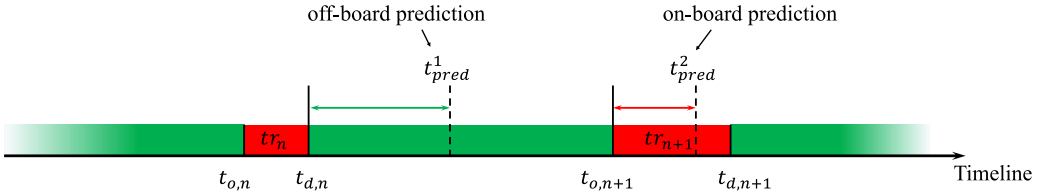


Fig. 2. Illustration of the off-board and on-board prediction problems. *Off-board* prediction is conducted in the interval between the last trip tr_n and the next trip tr_{n+1}; *on-board* prediction during a trip.

$n+1$, $(t_{o,n+1}, o_{n+1}, ?, w)$ and prediction time t_{pred} where $t_{o,n+1} < t_{pred} < t_{d,n+1}$, predict the destination d_{n+1} of the current trip.

Figure 2 illustrates the off-board and on-board prediction problems. The prediction time t_{pred} is the time when the prediction takes place. *Off-board* refers to the scenario that the individual has already finished his/her last trip (tr_n) when the prediction is conducted, while *on-board* means that the prediction is made during the individual's next trip (tr_{n+1}). Note that the prediction objective and available information are different in these two problems.

B. Model Framework

Figure 3 presents the proposed model framework, DeepTrip, consisting of three modules: travel feature extraction module, sequence pattern learning module, and prediction module.

- *Travel feature extraction module*. It takes individual trip sequences S as inputs and projects them into feature vector sequences. Different approaches are developed to extract discrete and continuous sequence features. Specifically, the discrete sequence attributes (i.e. origin, destination, and day of the week) are projected into vectors using the embedding function. An overlapped embedding model is proposed to project the continuous trip attributes (i.e. trip time), capturing both their categorical and numerical features.
- *Sequence pattern learning module*. It learns the sequential dependencies of individual trip sequences using a sequence-to-sequence structure with a multi-head attention mechanism. It processes the historical and real-time trip sequences separately (i.e. S_{V_h} and S_{V_c}), and projects them using the corresponding historical/real-time GRU networks. The multi-head attention layer looks up the historical trip sequence and finds the most relevant trip sequence $\boldsymbol{c}_n$ to the next trip prediction based on the current trip sequence information $\boldsymbol{h}_n$ and the prediction time point tp_τ. Finally, the selected historical trip sequences, real-time trip sequence, and the prediction time point are concatenated into a vector $[\boldsymbol{c}_n, \boldsymbol{h}_n, tp_\tau]$, serving as inputs to the prediction module.
- *Prediction module*. The prediction module includes a fully-connected neural layer with a softmax activation function. It takes as inputs the projected trip sequence vectors of all individuals and outputs the predicted information of the off-board of an individual.

The proposed DeepTrip structure provides a unified framework for dealing with both *on-board* and *off-board* prediction problems, by formulating model inputs differently. For the *off-board* prediction, the input for an individual u is the trip

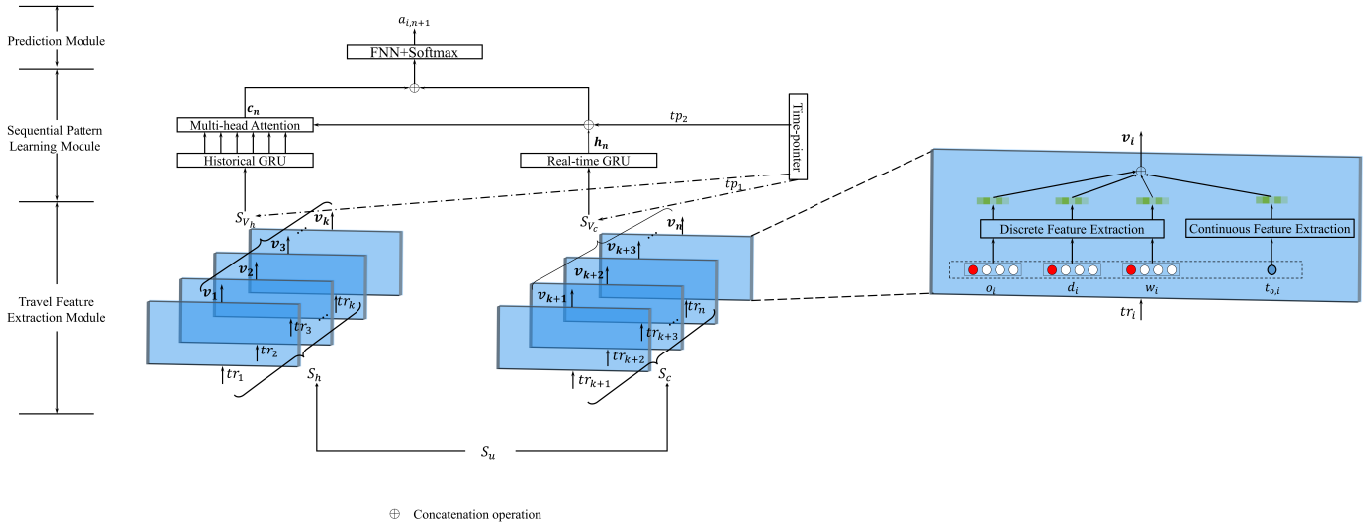


Fig. 3. Overview of the proposed framework. DeepTrip takes an individual's trip sequence S_u as input, divided into historical and recent sequences. Semantic features of the travel attributes and sequential dependency are captured by the travel feature extraction and sequential pattern modeling modules. The attributes of the next trip are predicted based on the combination of all the extracted features.

sequence S_u. For the *on-board* prediction, $o_{i+1}, t_{o,i+1}, w_{i+1}$ are known. Therefore, the input for individual u is $S_u = [(tr_1, o_2, t_{o,2}, w_2), \ldots, (tr_n, o_{n+1}, t_{o,n+1}, w_{n+1})]$.

C. Travel Feature Extraction Module

Trip attributes are represented by different data types, i.e. discrete and continuous. o, d, and w are categorical data (e.g. station number), while t_o is continuous. Existing studies represent the time t_o using hourly time intervals, e.g. 7:00-8:00 am [10], [22]. Let the time instance $t_i^0 \leq t_{o,i} \leq t_i^{24} \in \mathbb{T}$ be the starting time of trip i, which is between 0:00 AM and 24:00 PM of a day. The hourly interval representation of $t_{o,i}$ is:

$$hourly(t_{o,i}) = int\left(\frac{t_{o,i} - t_i^0}{t_i^{24} - t_i^0} \times 24\right) \tag{1}$$

where $hourly(t_{o,i})$ is the hourly representation of trip start time, and $int(\cdot)$ the floor function.

Although the hourly representation achieves good model performance, it suffers from potential risks of losing important features in temporal information. Theoretically, temporal information includes both categorical and numerical features. The categorical feature refers to the semantic relationship between time points regardless of the magnitude of the time difference.

For example, the time difference between 7:30 AM and 8:30 AM or 6:30 AM are both 1 hour. However, the mobility pattern at 8:30 AM is more similar to that at 7:30 AM compared to that at 6:30 AM, since 7:30 and 8:30 AM are in the morning peak while 6:30 AM is off-peak. The discrete representation can capture such categorical features well. However, it ignores the numerical feature of the temporal information and is limited in capturing chronological dependencies. For example, with the hourly interval representation, the interval difference between 8:59 AM and 9:01 AM is the same as that between 8:01 AM and 9:59 AM. However, conceptually the mobility pattern differences for these two cases could be substantial. The numerical feature is important for capturing temporal information, particularly for times around category boundaries. In the following, we develop 4 alternative strategies representing trip attributes.

1) *Embedding representation:* Following the word2vec idea [28], the categorical data is first represented using a one-hot format and then transformed into a dense vector using a one-layer linear neural layer. Compared to the direct one-hot representation, the Euclidean distance between dense vectors captures the semantic similarity and thus is more informative. For each trip attribute that has a discrete form, an embedding module is built to represent the raw data.

Continuous data can also be represented using the embedding function after discretization, e.g. hourly interval representation of t_o. After discretization, the temporal domain is transformed into a finite number of points with each point being well-trained during the training phase. Therefore, the model with such a representation can fully capture the semantic relationship between points. However, this representation has two important drawbacks: a) the discretization interval width is arbitrary and hard to determine. A large interval width may lose feature variability within an interval while a small one may decrease the signal-to-noise ratio (i.e. same features are likely to be separated into different intervals due to noise); b) the data near the interval boundary is discretized into different categories although they are very close.

The requiring strategies are developed to represent continuous trip attributes, especially the temporal information:

2) *Normalized representation:* the normalized representation of $t_{o,i}$ is:

$$normalize(t_{o,i}) = \frac{t_{o,i} - t_i^0}{t_i^{24} - t_i^0} \tag{2}$$

The normalized representation is a non-floor version of the hourly interval representation, in which the time t_o is transformed into a value between 0 and 1. The drawback of the normalized representation is that it only considers the numerical feature but lacks the categorical feature.

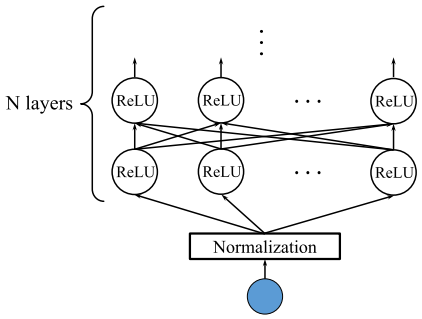


Fig. 4. Structure of the projection model.

3) *Projection representation:* The representation based on the projection model deals with the limitation of the previous approach by transforming the continuous data into a dense vector representation that captures both numerical and categorical features without discretized operations.

Figure 4 shows the projection model structure. It consists of two parts. First, time is transformed using the normalized representation. Second, a Multi-layer fully-connected Neural Network with a Rectified Linear Unit (ReLU) activation function takes the normalized value as input and projects it into a multi-dimension dense vector.

Theoretically, a neural network is a universal approximator [29], and thus capable of capturing both categorical and numerical features of the time information. The projection process in Figure 4 is mathematically written as:

$$t' = normalize(t) \tag{3}$$

$$\boldsymbol{out}_1 = ReLU(t'\boldsymbol{W}_{1\times K}) \tag{4}$$

$$\boldsymbol{out}_{i+1} = ReLU(\boldsymbol{out}_i \boldsymbol{W}_{K\times K}), \quad \text{for } i = 1, 2, \ldots, N \tag{5}$$

where t' is the normalized value of time t, $\boldsymbol{W}_{1\times K}$ and $\boldsymbol{W}_{K\times K}$ the weight matrices of stacked linear layers and $\boldsymbol{out}_i$ the output of the i^{th} linear layer. K is a hyper-parameter that controls the dimension of the output dense vector, and N is the number of neural layers.

Although the projection model captures both the categorical and numerical features of time, it may over-represent numerical features. In the problem studied in the paper, the numerical feature of time is informative when the time points are close indicating similar mobility patterns even if they may belong to different categories (e.g. the above-mentioned 8:59 AM and 9:01 AM case). However, when the time points are further, the mobility patterns are mostly dominated by categorical features other than the numerical difference. In addition, the projection model utilizes the original data which may lead to complex optimization hyper-planes. Thus it is prone to fall into local optimum solutions [30].

4) *Overlapped embedding representation:* To address issues in the projection representation, we propose a novel overlapped embedding model to capture the categorical and numerical features of time and automatically switch between them in the training process, to the one that better fits the nature of the mobility prediction problem. A sliding window, with an interval width l_w and a step l_s, is used to divide the day into T overlapped time intervals $\{I_1, I_2, \ldots, I_T\}$, where I_i is the i^{th} interval. Then, the time point t is represented as a vector

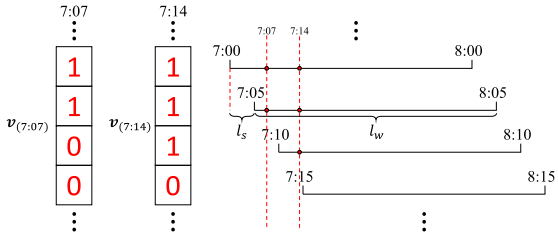


Fig. 5. Example of overlapped representation of time points with $l_s = 5$ minutes and $l_w = 1$ hour.

$\boldsymbol{V}$ with dimension T using the following rule:

$$v_i = \begin{cases} 1, & \text{if } t \text{ belongs to } I_i \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where v_i is the i^{th} element of $\boldsymbol{v}$.

Figure 5 shows an example of the overlapped embedding representation. The sliding window width is 1 hour and the step is 5 minutes. Given the T overlapped time intervals starting at 7:00 AM, the time point 7:07 AM is represented as [1, 1, 0, 0, ...] and 7:14 AM as [1, 1, 1, 0, ...]. Since 7:07 and 7:14 are very close to each other, only two cells of the represented vectors are different.

After encoding the continuous data using the overlapped representation, the encoded data is then fed into a linear layer (Figure 3) to transform it into a dense vector. The overlapped representation has three main advantages:

- It captures the mechanism of how the time variables capture mobility similarities. If the two time points are close, then most of the activated cells (i.e. cells with value 1) in the corresponding vectors overlap indicating similar features. As the distance between the two time points gets larger, the number of overlapped cells decreases until 0 (disjoint vectors). The categorical feature is gradually dominating the numerical feature, only the categorical feature contributes to the prediction.
- It is flexible and has less information loss compared to the commonly used one-hot encoding method. As discussed before, l_w in the one-hot encoding can not be too small (impact on signal-to-noise ratio), but loss of information if it is too large. In the overlapped representation, l_w and l_s are used to control the signal-to-noise ratio and information loss separately. Even a small value of l_w, e.g. 5 minutes can be used, without introducing much noise.
- It is more robust to noise compared to one-hot encoding, especially for small interval length l_w. For a feature variable with observation noise, the one-hot encoding is prone to assign it to the wrong category if the interval l_w is small. However, the overlapped representation is robust even with a small interval length (e.g. 5 minutes l_w instead of 1 hour in Figure 4).

In summary, all the categorical data, e.g. origin station, is represented by the embedding module. The continuous trip attributes are represented by either of the above four models. After the feature extraction, the original trip sequence is transformed into a sequence of dense vectors. In the case study in Section IV, we compare their performance in detail.

D. Sequence Pattern Learning Module

The essence of the next trip prediction is modeling the sequential pattern of an individual's trip sequence. Recurrent Neural Networks (RNN) are widely used to model sequential patterns of time series data. However, they suffer from the vanishing gradient problem for long sequence time series data [31]. We use the Gated Recurrent Unit (GRU) [32] to model the individual's trip sequence. GRU is a variant of Long-Short Term Memories (LSTM) and it has fewer parameters than LSTM and thus converges faster [33].

In terms of the information to be used for prediction, both the real-time and historical trip patterns are important, since an individual's next trip is highly related to recent trips and the long-term dependency of trips captures the weekly/monthly travel regularity. Given the different roles real-time and historical travel patterns may play in prediction, an individual's whole trip sequence is divided into two sub-sequences: real-time trip sequence $S_c = [tr_k, tr_{k+1}, \ldots, tr_n]$ and historical trip sequence $S_h = [tr_1, tr_2, \ldots, tr_{k-1}]$. These sequences are processed differently with all the real-time trip sequences S_c being used for prediction. However, instead of directly using the whole historical trip sequence S_h, only the ones that are most relevant to the current trip pattern are identified and used (contributing to the next trip prediction).

To model the interaction between historical/real-time trip sequences, we adopt a seq2seq model [34] coupled with multi-head attention and pairwise time-pointer mechanisms [35]. The seq2seq model uses an encoder-decoder structure consisting of two GRU networks. The encoder GRU (historical GRU network in Figure 3) takes S_{V_h} as input and outputs the hidden state vector $\boldsymbol{h}_i^h$ corresponding to each vector $\boldsymbol{v}_i$ in S_{V_h}. The decoder GRU (real-time GRU network in Figure 3) takes S_{V_c} as input. It not only outputs the hidden state vector step by step but also passes the hidden state vector to the encoder GRU at each step as a query to generate a context vector (i.e. a vector containing the relevant historical trip information) through a multi-head attention module.

1) Multi-Head Attention: The multi-head attention module consists of a set of single attention modules (Figure 6). The single attention module has been widely used in Natural Language Processing (NLP) applications, such as Machine Translation [36]. It generates a context vector $\boldsymbol{c}_n$ as a weighted sum of all the hidden states encoded by the encoder GRU (Equation 7). The weights capture the similarity between each encoder hidden state vector $\boldsymbol{h}_i^h$ and the last decoder hidden state vector $\boldsymbol{h}_n^c$, i.e. the hidden state vector corresponding to trip tr_n.

We adopt the *general* similarity form described in [34]. The attention weights are estimated through a Feedforward Neural Network (FNN) followed by a softmax activation function.

$$\boldsymbol{c}_n = \sum_{i=1}^{k-1} w_i \boldsymbol{h}_i^h \tag{7}$$

$$w_i = softmax(\boldsymbol{h}_i^h \boldsymbol{W}_A \boldsymbol{h}_n^c) \tag{8}$$

where $\boldsymbol{W}_A$ is a linear projection layer, $\boldsymbol{h}_i^h \boldsymbol{W}_A \boldsymbol{h}_n^c$ represents the *general* similarity. w_i is the multi-head attention weight of each historical hidden state vector $\boldsymbol{h}_i^h$ corresponding to $\boldsymbol{h}_n^c$. A larger weight value indicates a higher similarity. $k-1$ is the length of the historical trip sequence and $\boldsymbol{c}_n$ the context vector of the current time step.

The single attention module captures a limited semantic subspace of an individual's travel pattern (i.e. single trip attribute). A trip is characterized by multiple trip attributes. To capture the multi-aspect dependencies by trip attributes, we utilize a multi-head attention mechanism to model the joint interaction among different semantic subspaces of an individual's travel patterns. The multi-head attention module has N parallel single attention modules (or heads). The final context vector $\boldsymbol{c}_n'$ is generated as a weighted sum of all the context vectors from single attention modules:

$$\boldsymbol{r} = softmax(\boldsymbol{W}_q \cdot \boldsymbol{h}_{k-1}^h) \tag{9}$$

$$\boldsymbol{c}_n = \sum_{m=1}^{N} r_m \cdot \boldsymbol{c}_n^m \tag{10}$$

where $\boldsymbol{W}_q \in \mathbb{R}^{N \times d}$ represents the linear projection layer, $\boldsymbol{h}_{k-1}^h$ is the last hidden state of encoder GRU, N is the number of attention head, $\boldsymbol{c}_n^m$ is the context vector generated from mth head (attention module), r_m is the weight of $\boldsymbol{c}_n^m$.

A critical issue of the multi-head attention is that it leads to a redundancy problem, that is, all heads may eventually capture similar aspects of travel patterns. To avoid this problem, a penalty term is added to the final loss function. The penalty term penalizes the attention redundancy across different heads and forces different heads to focus on different travel patterns. It is calculated in two steps. First, for each head m, the average attention weight of each historical trip is calculated:

$$\delta_p^m = \frac{1}{M} \sum_{t=1}^{M} w_{t,p}^m \tag{11}$$

where M is the sequence length of the decoder GRU, $w_{t,p}^m$ the attention weight of the p^{th} encoder hidden state over the t^{th} decoder hidden state. $\boldsymbol{\delta}^m$ is the attention vector of the m^{th} head, where $\boldsymbol{\delta}^m = [\delta_1^m, \delta_2^m, \ldots, \delta_M^m]^T$. $\boldsymbol{\delta}^m s$ from each head are concatenated to construct a matrix $\boldsymbol{\Delta} = [\boldsymbol{\delta}^1, \boldsymbol{\delta}^2, \ldots, \boldsymbol{\delta}^N]$.

Then, the penalty term is calculated as follows:

$$\mathcal{L}_p = ||\boldsymbol{\Delta} \cdot \boldsymbol{\Delta}^T - \boldsymbol{I}||_F^2 \tag{12}$$

where $||\cdot||_F$ is the Frobenius norm and $\boldsymbol{I}$ an identity matrix with size $N \times N$. This formulation penalizes the similarity between attention vectors from different heads and forces the Euclidean norm of each attention vector to be close to 1.

Finally, the total loss function is:

$$\mathcal{L} = \lambda \mathcal{L}_{pred} + (1 - \lambda)\mathcal{L}_p \tag{13}$$

where, $\mathcal{L}_{pred}$ is the loss function of the mobility prediction task. λ is a hyper-parameter that controls the weight of each loss, with $\lambda \geq 0$.

2) Pairwise Time-Pointer: For the next-trip prediction problem, the prediction time information (the time when the prediction is made) is important to narrow down the solution space in order to improve the accuracy and efficiency of the prediction (as illustrated in Figure 1). It is challenging to

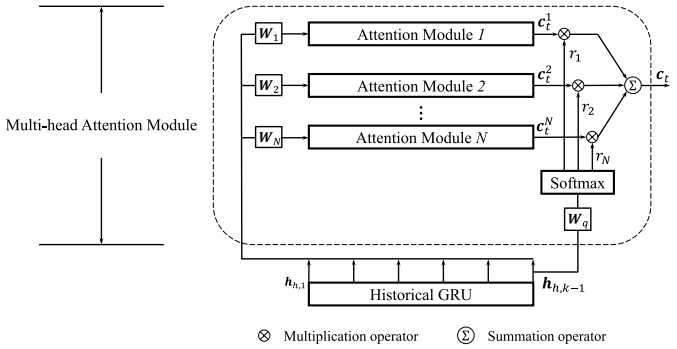


Fig. 6. Structure of the multi-head attention module.

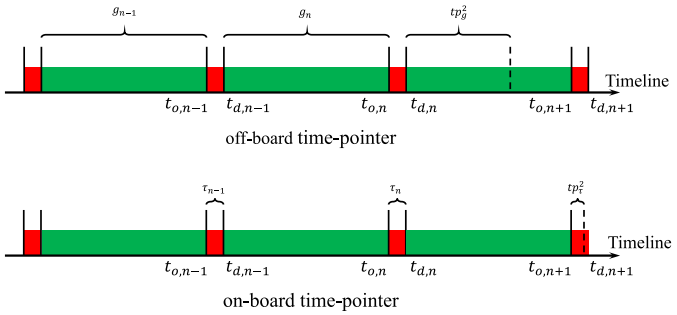


Fig. 7. Structure of pairwise time-pointer. The red bars represent the duration time of each trip, and the green bars the gap between two trips.

model the prediction time and incorporate it into the training process since the prediction time is dynamic and random (i.e. the next trip prediction can be made at any time in a day). Also, different prediction times provide different information for predicting the next trip given recent trips. We propose a pairwise time-pointer mechanism to enable the multi-head attention model to better select the relevant historical trips by making the best use of the prediction time. This mechanism also facilitates the model training process by systematically simulating the random prediction times in practice.

Figure 7 illustrates the pairwise time-pointer definition. The timeline shows the sequence of trips for an individual. The red section indicates the duration of a trip and the green section the gap between consecutive trips. The pairwise time-pointer is composed of two sequences of time information, including the trip gap time sequence and the randomized prediction time sequence.

The off-board time-pointer is $\{tp_g^1, tp_g^2\}$, where $tp_g^1 = [g_1, g_2, \ldots, g_n]$ is a sequence of gap times between consecutive trips and $g_i = t_{o,i} - t_{d,i-1}$. Each element of the gap time sequence tp_g^1 is normalized to $[0, 1]$ and concatenated with its corresponding trip vector $\boldsymbol{v}_i$ to formulate a new vector $(\boldsymbol{v}_i, g_i)$, which serves as the input to the GRU networks. The prediction time tp_g^2 is generated as $tp_g^2 = p \times (t_{o,n+1} - t_{d,n})$ where p is a random value from the uniform distribution $u(0, 1)$. It simulates a prediction conducted at a specific time point between $t_{d,n}$ and $t_{d,n+1}$. In the training process, the randomized prediction time tp_g^2 is concatenated to $\boldsymbol{h}_n^c$ and fed into the attention module.

By using the time-pointer, each training sample instance (i.e. trip sequence) is treated as conducting a prediction at the given prediction time tp_g^2. After convergence, a large amount of prediction time points and the corresponding trip sequences are simulated. Ultimately, the model will automatically learn the best use of the prediction time in predicting the next trip.

The on-board time-pointer $\{tp_\tau^1, tp_\tau^2\}$ in the sequence learning module behaves similarly to the off-board case. The only difference is that the on-board time-pointer captures the on-board trip attributes. Specifically, tp_τ^1 is the sequence of trip duration times $tp_\tau^1 = [\tau_1, \tau_2, \ldots, \tau_n]$, where $\tau_i = t_{d,i} - t_{o,i}$; and $tp_\tau^2 \in [t_{o,n+1}, t_{d,n+1})$ is randomly sampled to simulate the prediction time when an individual is *on-board* his/her $n+1$ trip.

E. Prediction

The output vector of the real-time GRU module $\boldsymbol{h}_n$, the context vector from the multi-head attention module $\boldsymbol{c}_n$, and the prediction time tp_2 are concatenated into a new vector $(\boldsymbol{h}_n, \boldsymbol{c}_n, tp_2)$. The prediction module takes as input the vector $(\boldsymbol{h}_n, \boldsymbol{c}_n, tp_2)$ and outputs the predicted next trip information. The prediction module is a NN network, consisting of a stack of FNN layers. The cross-entropy loss is used as the performance metric of the prediction task $\mathcal{L}_{pred}$:

$$\mathcal{L}_{pred} = -\log\left(\frac{e^{x[\text{True}]}}{\sum_j e^{x[j]}}\right) \tag{14}$$

where $e^{x[\text{True}]}/\sum_j e^{x[j]}$ denotes the output probability of the true class (e.g. the real origin station of the next trip) after softmax.

F. Training Algorithm

Three DeepTrip models are trained to predict o_{n+1}, d_{n+1}, and $t_{o,n+1}$ in the *off-board* prediction and one DeepTrip model to predict d_{n+1} in the *on-board* prediction. The training algorithm for the *on-board* and *off-board* predictions is the same except for the input trip sequence representations (as mentioned in section III-B). Algorithm 1 summarizes the *off-board* DeepTrip model training process. Note that in the

on-board case, tp_g^1 and tp_g^2 in Algorithm 1 are replaced by tp_τ^1 and tp_τ^2, respectively.

Algorithm 1 Training Algorithm for DeepTrip (Off-Board)

Input: Set of trip sequences U
Output: Trained model for a_i prediction
// constructing training set
$U^* \leftarrow \emptyset$
for each training sample S in U **do**
  calculate normalized tp_g^1;
  $S_h, S_c \leftarrow divide(S)$;
  $U^* \leftarrow (S_h, S_c, tp_g^1)$;
end for
// training phase
while $epoch < MAX\_EPOCH$ **do**
  $count \leftarrow 0$;
  $pred \leftarrow \emptyset$;
  for each instance (S_h, S_c, tp_g^1) in U^* **do**
    $S_{V_h} \leftarrow featureExtraction(S_h)$;
    $S_{V_c} \leftarrow featureExtraction(S_c)$;
    for each vector $\boldsymbol{v}_i$ in S_{V_h} **do**
      $\boldsymbol{v}_i \leftarrow (\boldsymbol{v}_i, g_i)$
    end for
    for each vector $\boldsymbol{v}_i$ in S_{V_c} **do**
      $\boldsymbol{v}_i \leftarrow (\boldsymbol{v}_i, g_i)$
    end for
    $HS_h \leftarrow encoder(S_{V_h})$
    $HS_c \leftarrow decoder(S_{V_c})$
    random sample $tp_g^2 \in [t_{d,n}, t_{o,n+1})$;
    $\boldsymbol{c}_n \leftarrow multiAttn(HS_c[-1], tp_g^2, HS_h)$;
    $\boldsymbol{p}_n \leftarrow concatenate(HS_c[-1], \boldsymbol{c}_n)$;
    $out \leftarrow softmax(FNN(\boldsymbol{p}_n))$;
    $pred$.append(out);
    $count \leftarrow count + 1$;
    if $count = BATCH\_SIZE$ **then**
      calculate $\mathcal{L} = \lambda \mathcal{L}_{pred} + \gamma \mathcal{L}_p$;
      $count \leftarrow 0$;
      $pred \leftarrow \emptyset$;
      if criteria is met **then**
        stop training; output trained model;
      else
        update model parameters θ;
      end if
    end if
  end for
end while

IV. Case Study

We evaluate the proposed framework using the automated fare collection (AFC) data from an urban metro system. The system currently consists of 11 railway lines, serving 91 heavy rail stations and 68 light rail stops. It serves over 5 million trips on an average weekday. For the urban heavy rail lines, trip transactions are recorded when passengers enter and exit the system, providing information about the tap-in and tap-out stations and corresponding timestamps. Individual trip data using the urban heavy railway (metro) from January 1st to March 31st in 2018 is used.

To validate the model performance, we select a random panel of 20,000 individuals who made at least 90 trips during the studied period (i.e. one trip per day on average). The trip attribute tuple includes origin station $o \in [0, 90]$, destination station $d \in [0, 90]$, day of the week $w \in [0, 6]$, and tap-in time $t_o \in [0, 23]$. These attributes are represented as categorical variables for comparison with the state-of-the-art models in the literature. Note that for DeepTrip, we use the proposed overlapped embedding to represent t_o. Each trip can be characterized by a trip attribute tuple (o, d, t_o, w), and the individual's trip sequence is captured by these tuples in chronological order.

To increase the size of the training sample, We utilize a sliding window of width n to generate the trip sequence sample instances from the individual's trip sequence. Specifically, given an individual's trip sequence $S_u = [tr_1, tr_2, \ldots, tr_N]$, each trip sequence sample S_{sample} is generated as $S_{sample} = [tr_i, tr_{i+1}, \ldots, tr_{i+n-1}]$ with i ranging from 1 to $N - n + 1$. Accordingly, an individual with N trip records results in $N - n + 1$ trip sequence sample instances. For each sample instance, the prediction time information is generated based on the random-sampling method, i.e. randomly choosing a time point between the last two trips.

These trip sequence sample instances are used as model input in the case study. We use a window width $n = 70$ as 95.2% of the individuals in the panel have less than 70 trips per month. The width 70 adequately captures the periodic features of an individual's travel behavior (both weekly and monthly patterns). Finally, 1,386,872 trip sequence sample instances are generated. Samples generated from 80% of the individuals are used for training and the rest 20% for testing. For each sample, the travel attributes of the last trip (i.e. o, d, and t_o) are set as the prediction target.

The experiments are designed to validate the proposed DeepTrip model performance by comparing the results with the results from state-of-the-art statistical and deep learning models, as well as explore the impact of feature representation and prediction time information on the DeepTrip model prediction performance. We use the fraction of the correctly predicted trip attributes as the performance metric to evaluate the model prediction accuracy:

$$Accuracy = \frac{N_{right}}{N_{total}} \tag{15}$$

where N_{right} represents the number of sample instances with the correct prediction, and N_{total} denotes the total number of sample instances.

A. Model Validation and Performance Comparison

We build 3 DeepTrip models for o, d, and t_o prediction, respectively. The model structures are the same, the only differences are the output and the training target.

We compare the proposed DeepTrip model with 4 baseline models in the literature: the Mobility Markov Chain model [15], the Mobility N-Gram model [9], the DeepMove [10] model, and the MobTCast [24] model. The Markov

TABLE II
MODEL CHARACTERISTICS

Model	Approach	Prediction Target	Temporal Representation	Prediction Time
Markov	Statistical	Next Location	Index	×
N-gram	Statistical	Next Trip	Index	×
DeepMove	Deep Learning	Next Location	One-hot	×
MobTCast	Deep Learning	Next Location	One-hot	×
DeepTrip	Deep Learning	Next Trip	Overlapped	✓

Chain and N-Gram models are classic statistical models, while DeepMove and MobTCast are deep learning-based models. The main differences between these models are summarized in Table II. The prediction target represents the model′s prediction outputs, either the next trip (multi-attributes) or location only. The temporal representation is the data representation method for the temporal information of trips. The prediction time represents the model's ability to capture the prediction time information and make predictions at arbitrary times.

Besides evaluating the whole test set (i.e. overall evaluation), each model is verified under 3 more scenarios, which are defined based on the length of the gap time g_{n+1} between the start time of the predicted trip and the previous trip end time.

- Short-term prediction: $g_{n+1} < 2h$;
- Medium-term prediction: $2h \leq g_{n+1} < 24h$;
- Long-term prediction: $g_{n+1} \geq 24h$.

To make a consistent comparison across the different models, the following settings are used:

a) Mobility Markov Chain model. The first-order Markov Chain model showed the best prediction performance among other Markov Chain model structures;
b) Mobility N-gram model. The settings of the mobility N-gram model were identical to the ones used by the authors [9];
c) MobTCast. MobTCast is a transformer-based model, 2 layers of transform blocks are used which achieve the best performance;
d) DeepMove. The hyper-parameter settings are shown in Table III. For a fair comparison with the DeepTrip model, added the gap time information between consecutive trips tp_1 into the DeepMove model.
f) DeepTrip. The *off-board* DeepTrip model is used in this evaluation since its input trip tuple has the same formulation as the above models. The DeepTrip model used the same hyper-parameter settings as the DeepMove (Table III). The results of DeepTrip in Table IV is the mean value of the accuracy under the prediction time ranging from 1% to 100% of the gap time g_{n+1}.

Table IV compares the prediction results of the next trip attributes by prediction gap times. Generally, all models perform better for medium-term prediction than short- or long-term predictions. This could be attributed to the travel regularity and training sample size. The medium-term scenario exhibits the best results since most commuting trips are within 24 hours from the last trip, which is more regular and provides large samples for the model to learn.

TABLE III
SETTINGS FOR DEEPMOVE AND DEEPTRIP

Settings	Value	Settings	Value
optimizer	Adam	embedding size of o	20
learning-rate	0.0009	embedding size of d	20
gradient clip	5.0	embedding size of t	10
GRU hidden size	60	embedding size of w	5
dropout	0.2	batch size	50

TABLE IV
PERFORMANCE EVALUATION

		Markov	N-gram	DeepMove	MobTCast	DeepTrip
t_o	Overall	37.2%	39.9%	44.5%	45.2%	**47.3%**
	0-2h	23.0%	25.9%	33.3%	32.9%	**38.6%**
	2-24h	41.1%	43.7%	47.4%	48.1%	**49.3%**
	24h-	24.6%	27.6%	29.6%	30.9%	**31.5%**
o	Overall	58.5%	61.7%	79.4%	79.2%	**81.3%**
	0-2h	35.6%	31.1%	82.0%	75.7%	**82.5%**
	2-24h	67.6%	63.6%	79.8%	79.5%	**80.0%**
	24h-	47.1%	54.5%	59.5%	61.2%	**61.3%**
d	Overall	49.7%	53.4%	61.6%	61.6%	**62.6%**
	0-2h	36.4%	46.4%	47.8%	47.2%	**49.9%**
	2-24h	53.9%	63.6%	64.2%	64.3%	**64.5%**
	24h-	33.1%	34.3%	39.9%	39.9%	**40.1%**

Among those models, the deep learning-based models outperform the statistical-based methods by around 10% for predicting o and d and 5% for t_o. Surprisingly, The MobTCast model exhibits marginal performance improvement compared to the GRU-based DeepMove model in the next trip prediction. This is different from the model comparison results for the GPS-based location prediction problem in [24]. It indicates that a large model (MobTCast) may not necessarily perform better than a small model (DeepMove). The possible reasons could be two-fold: 1) the MobTCast model size (or capacity) is much redundant for the studied problem that could be prone to the overfitting issue; 2) the studied problem and dataset have regular mobility patterns, which results in low model uncertainty and thus favor a small deep learning model. We provide further experimental analysis and literature evidence to support these arguments in the supplementary information file.

The proposed DeepTrip model outperforms the DeepMove model, increasing the accuracy by around 2%. These results indicate that incorporating the prediction time information plays an important role in predicting individual mobility. Also, it can be seen that the improvement in the accuracy of predicting t_o (around 3%) is higher than the other two trip attributes. That is because the prediction time information provides a hard constraint on the candidate tap-in times given the fact that the next tap-in event will happen later than the prediction time.

B. Ablation Experiment

To evaluate the effectiveness of each module contained in the proposed DeepTrip model, we conduct the ablation

TABLE V
ABLATION EXPERIMENT RESULT

	o	d	t_o
GRU	78.1%	58.9%	41.4%
Seq2seq	78.1%	59.2%	41.6%
Seq2seq+Attn	79.4%	61.6%	44.5%
DeepTrip	**81.3%**	**62.6%**	**47.3%**

experiment which evaluates the model performance by gradually incorporating the modules:

- *GRU*: use one single GRU.
- *Seq2seq*: use the seq2seq structure which contains an encoder GRU and a decoder GRU.
- *Seq2seq+Attn*: Seq2seq structure combined with the attention module.
- *DeepTrip*: incorporates the random-sampling training method into Seq2seq+Attn, which is the original form of DeepTrip.

Table V summarizes the ablation experiment results. In general, the DeepTrip model outperforms all its variants in predicting the next trip attributes. The GRU model yields the worst performance, as inputting the whole sequence into one single GRU makes it indiscriminate of the history and recent travel patterns which would mislead the model in sequential pattern learning. The prediction performance slightly improves when the Seq2seq structure is used which separates the history and recent travel pattern information. The incorporation of the attention module boosts the performance (1.3% for o, 2.4% for d, and 2.9% for t_o) over Seq2Seq. The reason is that the attention module offsets the GRU's poor ability to handle long sequences. The DeepTrip model achieves the best prediction performance by taking advantage of the Seq2seq+Attn model and using the prediction time information. Incorporating the prediction information improves the model performance by 1-3% for each attribute compared to not. In all, the results highlight the importance of the attention mechanism and the prediction information in the DeepTrip model.

C. Impact of Data Representation

The data representation (discrete or continuous) is important for prediction. We evaluate the impact of the different representation approaches (normalized, embedding, projection, and overlapped) on the DeepTrip model prediction accuracy. The first three approaches are commonly used in the literature and the overlapped model is proposed to capture both the categorical and numerical features of the continuous variable (i.e. t_o).

Table VI summarizes the prediction results with the different representation models. The *With_pred* scenario incorporates the prediction time information, while the *No_pred* does not. The o, d_{off}, and t_o are the origin station, destination station, and tap-in time in the *off-board* prediction, while the d_{on} is the destination station in the *on-board* prediction.

The results show that the models using the prediction time information perform better than those without using such information, particularly for predicting the next trip time t_o (3%) and next day of travel w (8%). Compared to the *off-board* prediction, the prediction time information contributes much less to the *on-board* prediction.

TABLE VI
PREDICTION PERFORMANCE OF THE DEEPTRIP MODEL WITH DIFFERENT REPRESENTATIONS OF t_o

	No_pred			
	Normalized	Embedding	Projection	Overlapped
o	79.1%	**79.4%**	**79.4%**	79.1%
t_o	40.1%	44.5%	43.7%	**44.8%**
d_{off}	61.3%	**61.6%**	61.3%	61.5%
d_{on}	69.2%	70.7%	70.8%	**71.0%**
w	80.8%	81.5%	**82.1%**	81.7%
	With_pred			
	Normalized	Embedding	Projection	Overlapped
o	80.5%	81.2%	**81.4%**	81.3%
t_o	42.1%	47.1%	46.1%	**47.3%**
d_{off}	61.9%	62.5%	**62.7%**	62.6%
d_{on}	69.9%	71.3%	71.3%	**71.5%**
w	88.0%	89.9%	**90.2%**	90.1%

Comparing different representation models, the normalized representation performs the worst due to its poor ability in modeling the categorical feature of the temporal information. The overlapped representation performs the best in predicting t_o and d_{on}. These two attributes are sensitive to the temporal information representation as the model predicts exactly the time point t_o and the on-board destination prediction d_{on} is influenced by the departure time. For t_o prediction, an inappropriate representation of time may predict two closed time points to two different time intervals. For d_{on} prediction, a slight difference in the departure time may probably result in different destinations. The proposed overlapped representation captures both the categorical and numerical features of the temporal information that better fits the next trip prediction problem characteristics. However, the overlapped representation performs slightly worse in predicting o and d_{off} than the projection and embedding representations. This is partly because the off-board prediction of origin/destination stations is less sensitive to temporal information and thus hardly benefits from the overlapped representation. Instead, it may get worse due to the fact that more parameters are introduced.

D. Impact of the Prediction Time Information

The contribution of incorporating prediction time information varies depending on the prediction context. For example, if the prediction is conducted just few minutes after the end of the last trip, the prediction time adds little information and hence, does not improve the accuracy of the prediction of the next trip. However, if the prediction time is far from the last trip ending time, it benefits the prediction by filtering out certain candidate trips.

We assess the impact of prediction time information on model performance as follows: First, we train the DeepTrip models for all trip attributes for the *on-board* and *off-board* problems. Then, we simulate the actual prediction for different

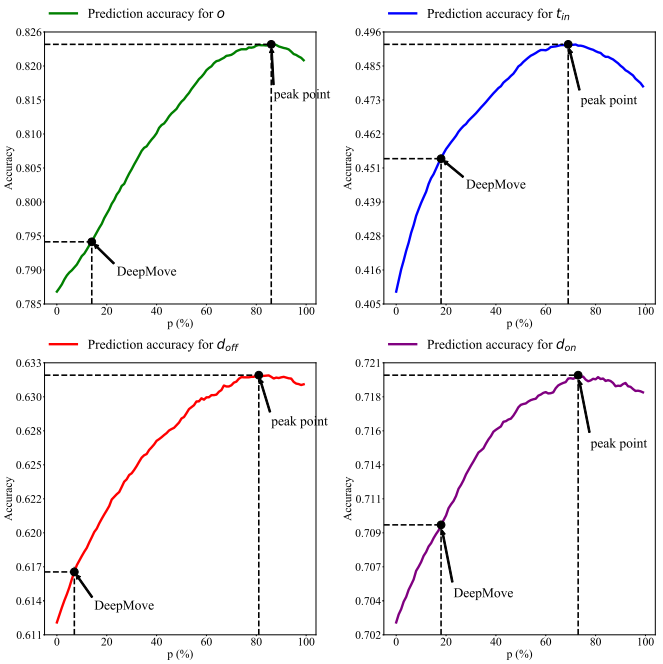


Fig. 8. Prediction performance as a function of different percentages of prediction time. d_{off} and d_{on} represent the prediction of d in off-board and on-board scenarios, respectively.

values of $tp_g^2 = p \times (t_{o,n+1} - t_{d,n})$ and $tp_\tau^2 = p \times (t_{d,n+1} - t_{o,n+1})$, with p ranging from 0% to 100%. $p = 0\%$ means that the prediction time is equal to the end of the n^{th} trip, while $p = 100\%$ means that the prediction time is equal to the start time of the next trip (i.e. $n + 1th$ trip). Figure 8 shows the prediction accuracy for o, d, and t_o predictions as a function of the prediction time. We also trained the corresponding DeepMove models as a benchmark for the analysis. The DeepMove prediction accuracy is also shown in Figure 8.

Figure 8 shows that the prediction accuracy of DeepTrip keeps increasing with the increase of p, i.e. the time gap between the prediction time point and the previous trip ending time (off-board prediction) or the current trip starting time (on-board prediction). The DeepTrip model performs slightly worse than the DeepMove model in the beginning when the prediction is made immediately after the last trip is finished (or tap-in). A possible reason could be that the prediction time hardly provides useful information for the next trip prediction when it is close to the end of the last trip. In such cases, from an algorithmic perspective, the DeepTrip model trained using the less important prediction time information is less efficient than the DeepMove model with no such information.

The DeepTrip model outperforms the DeepMove model when the p exceeds 20%, reaches its peak when the p is around 70-86%, and then starts to drop after that. The performance degradation for p close to 1 can be attributed to the fact that the correct next trip may be filtered out if the prediction time is too close to the t_b^{n+1}. Figure 9 provides an example to illustrate the potential impact of prediction time information. It shows the actual distribution of the start time of the next trip. When the prediction time is close to the last trip (i.e. t_{pred}^1), the DeepTrip algorithm could filter out impossible candidate trips without affecting the likelihood of the next trip.

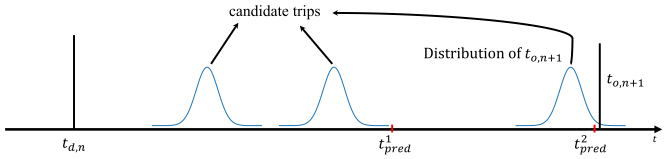


Fig. 9. Illustration of the impact of prediction time information.

However, if the prediction time is t_{pred}^2, the next trip $trip_{n+1}$ would be filtered out given its small probability to happen, which may lead to a false prediction result.

V. Conclusion

This paper presents an individual next trip prediction framework. The deep learning-based structure facilitates capturing multi-dimensional mobility patterns. The proposed time-pointer mechanism and random-sampling training algorithm simulate the prediction time information during the training phase. Compared with both classical machine learning models and state-of-the-art deep learning models, the proposed framework improves performance. The prediction accuracy is around 81% for origin locations, 62% for destination locations, and 47% for trip origin starting times, a 2-3% increase from the current SOTA model. This indicates that incorporating prediction time information is essential for the next trip prediction. We also propose variants of trip attribute representation to broaden the generalization ability of the framework. Four alternative strategies are proposed to extract the temporal information from different representation forms and evaluated.

The case study based on real-world data provides valuable insights. The prediction accuracy increases as the prediction time approach the departure time of the next trip. Our empirical results indicate that the best prediction performance could be obtained when the prediction time is around 80% of the off-board/on-board time. The model can also predict the next trips happening across days, not necessarily daily.

There are a number of promising directions for further research. Developing models that could capture the domain-specific information (e.g., the spatial similarity of activities) is an interesting direction. The proposed framework mainly focuses on frequent passengers since the advantage of the deep learning-based framework is based on mining complex long-term mobility features. Predicting infrequent users' mobility patterns is a challenging problem that may be worth exploring.

Acknowledgment

The authors would like to acknowledge the metro agency for providing the data support.

References

[1] N. Keyfitz, "Individual mobility in a stationary population," *Population Stud.*, vol. 27, no. 2, p. 335, Jul. 1973.

[2] J. L. Toole, M. Ulm, M. C. González, and D. Bauer, "Inferring land use from mobile phone activity," in *Proc. ACM SIGKDD Int. Workshop Urban Comput.*, Aug. 2012, pp. 1–8.

[3] M. W. Horner and M. E. O'Kelly, "Embedding economies of scale concepts for hub network design," *J. Transp. Geography*, vol. 9, no. 4, pp. 255–265, Dec. 2001.

[4] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala, "Bluetooth and WAP push based location-aware mobile advertising system," in *Proc. 2nd Int. Conf. Mobile Syst., Appl. Services (MobiSys)*, 2004, pp. 49–58.

[5] S. Çolak, A. Lima, and M. C. González, "Understanding congested travel in urban areas," *Nature Commun.*, vol. 7, no. 1, p. 10793, Mar. 2016.

[6] F. Calabrese, G. Di Lorenzo, and C. Ratti, "Human mobility prediction based on individual and collective geographical preferences," in *Proc. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2010, pp. 312–317. [Online]. Available: http://hdl.handle.net/1721.1/101713http://creativecommons.org/licenses/by-nc-sa/4.0/, doi: 10.1109/ITSC.2010.5625119.

[7] Z. Ma and H. N. Koutsopoulos, "Optimal design of promotion based demand management strategies in urban rail systems," *Transp. Res. C, Emerg. Technol.*, vol. 109, pp. 155–173, Dec. 2019.

[8] G. Gidófalvi and F. Dong, "When and where next: Individual mobility prediction," in *Proc. 1st ACM SIGSPATIAL Int. Workshop Mobile Geographic Inf. Syst., (MobiGIS), Conjunct 20th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst., (GIS)*, Nov. 2012, pp. 57–64.

[9] Z. Zhao, H. N. Koutsopoulos, and J. Zhao, "Individual mobility prediction using transit smart card data," *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 19–34, Apr. 2018.

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

[11] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[12] X. Lu, E. Wetter, N. Bharti, A. J. Tatem, and L. Bengtsson, "Approaching the limit of predictability in human mobility," *Sci. Rep.*, vol. 3, no. 1, Oct. 2013.

[13] V. Kulkarni, A. Mahalunkar, B. Garbinato, and J. Kelleher, "Examining the limits of predictability of human mobility," *Entropy*, vol. 21, no. 4, p. 432, Apr. 2019.

[14] D. Ashbrook and T. Starner, "Learning significant locations and predicting user movement with GPS," in *Proc. Int. Symp. Wearable Comput. (ISWC)*, Jan. 2002, pp. 101–108.

[15] S. Gambs, M.-O. Killijian, and M. N. Del Prado Cortez, "Next place prediction using mobility Markov chains," in *Proc. 1st Workshop Meas., Privacy, Mobility*, Apr. 2012, pp. 1–6.

[16] A. Asahara, K. Maruyama, A. Sato, and K. Seto, "Pedestrian-movement prediction based on mixed Markov-chain model," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2011, pp. 25–33.

[17] R. Kohavi and M. Sahami, "Error-based and entropy-based discretization of continuous features," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining (KDD)*. Portland, OR, USA: AAAI Press, 1996, pp. 114–119.

[18] W. Mathew, R. Raposo, and B. Martins, "Predicting future locations with hidden Markov models," in *Proc. ACM Conf. Ubiquitous Comput.*, Sep. 2012, pp. 911–918.

[19] C. Zhang, K. Zhang, Q. Yuan, L. Zhang, T. Hanratty, and J. Han, "GMove: Group-level mobility modeling using geo-tagged social media," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1305–1314.

[20] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "WhereNext: A location predictor on trajectory pattern mining," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jun. 2009, pp. 637–645.

[21] H. P. Hsieh, C. T. Li, and X. Gao, "T-Gram: A time-aware language model to predict human mobility," in *Proc. 9th Int. Conf. Web Social Media (ICWSM)*, 2015, pp. 614–617.

[22] A. Rossi, G. Barlacchi, M. Bianchini, and B. Lepri, "Modelling taxi drivers' behaviour for the next destination prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 2980–2989, Jul. 2020.

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

[24] H. Xue, F. Salim, Y. Ren, and N. Oliver, "MobTCast: Leveraging auxiliary trajectory forecasting for human mobility prediction," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–12.

[25] S. Tao, J. Jiang, D. Lian, K. Zheng, and E. Chen, "Predicting human mobility with reinforcement-learning-based long-term periodicity modeling," *ACM Trans. Intell. Syst. Technol.*, vol. 12, no. 6, pp. 1–23, Dec. 2021.

[26] J. Zhao, L. Zhang, J. Ye, and C. Xu, "MDLF: A multi-view-based deep learning framework for individual trip destination prediction in public transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 13316–13329, Aug. 2022.

[27] F. Zhou, Y. Dai, Q. Gao, P. Wang, and T. Zhong, "Self-supervised human mobility learning for next location prediction and trajectory classification," *Knowl.-Based Syst.*, vol. 228, Sep. 2021, Art. no. 107214.

[28] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1–6.

[29] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[30] G. Ke, Z. Xu, J. Zhang, J. Bian, and T.-Y. Liu, "DeepGBM: A deep learning framework distilled by GBDT for online prediction tasks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 384–394.

[31] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[32] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.

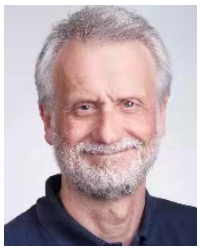
[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.

[35] C. Tao, S. Gao, M. Shang, W. Wu, D. Zhao, and R. Yan, "Get the point of my utterance! Learning towards effective responses with multi-head attention mechanism," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 4418–4424.

[36] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent. Conf. Track*, 2015, pp. 1–15.

Pengfei Zhang received the M.S. degree in control engineering from Hohai University in 2017 and the Ph.D. degree in traffic information engineering and control from the South China University of Technology in 2021. From 2019 to 2020, he was a Visiting Ph.D. Student with the NU-MIT Transit Group, Northeastern University, Boston, MA, USA. He is currently an Assistant Researcher with the Institute of Physics, Henan Academy of Sciences, Zhengzhou, China. His research interests include deep learning in transportation, individual mobility modeling, computer vision, and transportation mode choice.



Haris N. Koutsopoulos is currently a Professor with the Department of Civil and Environmental Engineering, Northeastern University, Boston, MA, USA. His current research interests include the use of data from opportunistic and dedicated sensors to improve planning, operations, monitoring, and control of urban transportation systems. He was a recipient of the Traffic Simulation Lifetime Achievement Award.

Zhenliang Ma is currently an Assistant Professor of road traffic engineering with the KTH Royal Institute of Technology. His research interests include statistics, machine learning, computer science based modeling, simulation, optimization, and control within the framework of selected mobility-related complex systems, which are: intelligent transport systems (traffic public transport/rails) and personal information systems (transport/energy).