LRS4DP: Location Recommendation System for Destination Prediction

1st Bing Zhao
Institute for Infocomm Research
*A*STAR*
Singapore
zhao_bing@i2r.a-star.edu.sg

2nd Haodong Sun
Beijing Key Laboratory of Traffic Engineering
Beijing University of Technology
Beijing, China
sunhaodong@emails.bjut.edu.cn

3rd Wee Siong Ng
Institute for Infocomm Research
*A*STAR*
Singapore
wsng@i2r.a-star.edu.sg

4th Roy Ka-Wei Lee
Information Systems Technology and Design
Singapore University of Technology and Design
Singapore
roy_lee@sutd.edu.sg

5th Yanyan Chen
Beijing Key Laboratory of Traffic Engineering
Beijing University of Technology
Beijing, China
cdyan@bjut.edu.cn

***Abstract*—Destination prediction based on the partial trajectory of a moving vehicle is vital for urban mobility applications. Recent research efforts focus on improving the prediction accuracy by incorporating more spatio-temporal semantics through complex model architectures, which inevitably impact the generalization and scalability due to ad-hoc hyper-parameters and heavier computations. In the present study, we propose a novel Location Recommendation System for Destination Prediction, LRS4DP. Through an integrated design of several technologies (map-matching, deep learning and recommender system), LRS4DP provides an end-to-end solution for destination prediction based on input trajectories and road network configurations. By adopting a node-based spatial discretization scheme through map-matching, LRS4DP is able to adapt according to the local road network density and generalize to different urban layouts. As compared to the state-of-the-art algorithms, our proposed Top-K formulation based on individual road nodes leads to fundamentally better spatial precision and prediction accuracy even with simple model architectures. We further designed the offline training and online serving as a location recommendation system to achieve better scalability and flexible trade-off between performance and run-time. The experimental evaluation of two real-world taxi datasets demonstrates the generalization of LRS4DP under different urban scales and layouts. The LRS4DP framework is also generically applicable for location prediction tasks (e.g., next location and passing-by location predictions) and capable to support various downstream transportation and location-based service applications.**

***Index Terms*—transportation, destination, trajectory, recommender system**

I. Introduction

The growing usage of Global Navigation Satellite System (GNSS) devices is generating massive trajectory data with increasing spatial and temporal resolutions. This leads to the recent flourish of data-driven transportation applications, such as flow prediction for rail transit systems with sparse origin-destination pairs [1], falsified vehicle trajectory detection for the cyber security of vehicle-to-infrastructure systems [2], generative trajectory modeling for better data privacy, and marine vessels destination prediction at a global scale [3].

The prediction of destinations based on real-time partial trajectories is of fundamental importance for many location-based services and applications. With the knowledge of potential destinations, we can achieve more personalized advertisement [4]–[6], effective traffic flows management [7], [8] and efficient vehicle dispatching [9], etc. With the development of autonomous vehicles, people may eventually spend less time driving but more time considering where to go. Thus, destination prediction formed the cornerstone for urban mobility and is a key factor for the business success of location-based service providers.

The obvious spatio-temporal travel patterns (e.g., popular origin-destination pairs, weekly trends, etc.) provide fundamental support to the development of memory-based learning algorithms, such as Bayesian inference models [10], Markov models [11], trajectory similarity search [12], and symbolic trajectory databases specially designed for spatio-temporal query [13]. However, with a more complex context or larger state space, the performance of memory-based learning is limited due to the increasing sparsity of the transition probability matrix. Thus, more recent efforts focus on representation learning, especially the deep learning-based trajectory modeling. Recurrent neural network (RNN) [14], [15] and convolutional neural network (CNN) [6], [16] based models fit in naturally due to their ability to capture sequential and spatial patterns, respectively. With deep representation learning, we can incorporate more complex spatio-temporal semantics into the prediction framework [17], [18], and multi-task learning (with simultaneous travel time prediction) can also boost the prediction accuracy [8]. However, such models require ad-hoc pre-processing and hyper-parameter tuning when applied to different road network and traffic distributions.

Although the coordinates of destinations are continuous quantities that fit naturally into regression models [17], it is semantically more meaningful to formulate destination prediction as a classification problem returning the top can-

didate locations. Considering two points located close to each other but on opposite sides of a two-way road, they may be physically far apart in terms of connectivity. The classification formulation also allows the mining of diverse interests of the travelers and provides more generic support for various downstream applications. However, the resolution of spatial discretization determines the spatial precision, and increasing resolution leads to an exponential expansion of the candidate pool and associated computation [6], [16].

In the present study, we propose a novel Location Recommendation System for Destination Prediction, LRS4DP. Through an integrated design of several technologies (map-matching, deep learning and recommender system), LRS4DP provides an end-to-end solution for destination prediction based on input trajectories and road network configurations. By adopting a node-based spatial discretization scheme through map-matching, LRS4DP is able to adapt according to the local road network density and generalize to different urban layouts. Comparing to the state-of-the-art algorithms, our proposed Top-K formulation based on individual road nodes leads to fundamentally better spatial precision and prediction accuracy even with simple model architectures. We further designed the offline training and online serving as a location recommendation system to achieve better scalability and flexible trade-off between performance and run-time. The experimental evaluation of two real-world taxi datasets demonstrates the generalization of LRS4DP under different urban scales and layouts. The LRS4DP framework is also generically applicable for location prediction tasks (e.g., next location and passing-by location predictions) and capable to support various downstream transportation and location-based service applications.

The remaining parts of this paper are organized as follows. Section 2 introduces the preliminaries for understanding the nature of trajectory data, spatial discretization, and modeling for destination prediction; Section 3 reviews deep learning-based sequential and convolutional destination prediction models; Section 4 provides details on the design of LRS4DP framework; Section 5 describes the experimental design and numerical implementation; Section 6 reports the experimental results to validate the performance and demonstrate the generalization capability of LRS4DP; and Section 7 concludes the present study.

II. Related Works

This section reviews the existing deep learning-based trajectory models for destination prediction. The central idea is to encode the variable-length partial trajectory sequence into a dense vector to alleviate data sparsity, especially when considering complex contextual information. Here, we divide the existing models into two main categories: sequential and convolutional models.

A. Sequential Models

ECML/PKDD'15 held a Kaggle competition on destination prediction of taxi trips in Porto, Portugal. Without spatial discretization, the winners encoded the partial trajectories (represented by the raw coordinates of the first and last five GPS points) into embedding vectors by testing different encoders, including Multi-layer Perceptron (MLP), bidirectional RNN (BRNN), BRNN with time window and memory network [19]. The contextual features (e.g., taxi ID, hours of the day, day of the week, etc.) were also embedded and concatenated with the partial trajectory embedding before being passed into the final fully-connected layer for classification. The classification candidates were cluster centers derived beforehand by applying a mean-shift clustering algorithm on historical destinations. The final destination was predicted as a weighted average of the center coordinates as below,

$$\widehat{y} = \sum_{i=1}^{C} p_i c_i \tag{1}$$

where C represents the number of clusters. The accuracy of destination prediction was measured by the mean Haversine distance between predicted and ground-truth destination coordinates. The authors concluded that their best model was BRNN with time window, given their clustering algorithm returned 3,392 cluster centers. Based on similar formulation, [20] archieved better accuracy by employing an ensemble model based on Support Vector Regression (SVR) and Deep Belief Network (DBN). So far, there were no discussions on the selection criterion and sensitivity to the pre-clustering step in those studies. [19].

With spatial discretization by road networks, the multi-scale topological structure of the road network is captured automatically through the training of location embedding vectors. [17] proposed a Trembr model to first learn an embedding for each road segment (Road2Vec) through multi-task learning by jointly predicting whether two road segments occur in the same trajectory and whether two road segments belong to the same road type. Trembr then utilize the Road2Vec model to encode the trajectory through an RNN-based architecture and predict the destination as regression tasks with two linear models for latitude and longitude separately. However, the training of Road2Vec model requires feature extraction and labeling by sliding window with a customized window size, and the authors only trained and tested their model under the condition that 80% of the full trjectory was known.

As a design choice, some studies employ spatial discretization by structured grids and express a trajectory as a sequence of grid IDs. Under this scheme, [5] attempted to predict the visiting probabilities of multiple destinations by proposing a one-step transition model. They formulated their model to output the transition probabilities towards the following grid. They estimated the visiting probabilities of candidate destinations through stochastic sampling until reaching a maximum step size. However, the algorithm has two practical drawbacks: (i) it did not consider temporal features and their effect on the transition probabilities at different steps, and (ii) the maximum step size limited the prediction horizon and remained a hyper-parameter to be tuned.

As details within each grid are lost, the multi-scale characteristics of trajectories require explicit consideration through hierarchical models that extract and merge features in different spatial granularities [21]. With added considerations of the multi-scope characteristics, [15] proposed an H-LATL model architecture for destination prediction by (i) introducing a Location-Aware Attention mechanism following the LSTM block; (ii) incorporating spatial-temporal intervals between locations into the Trajectory-LSTM block; and (iii) employing a hierarchy of models for various spatial granularities, each with its own customized neural network layers (e.g., additional dimension reduction for longer sequences under the finer granularity). Despite the performance gain validated with two taxi datasets, the practical implementation of their algorithm requires tuning of additional hyper-parameters associated with the hierarchical architecture, and the scalability could be a practical issue given its heavy model architecture.

B. Convolutional Models

Convolutional models extract the spatial patterns of trajectories by considering them as 2D images. However, the spatial resolution required varies due to the uneven road network density and the multi-scale characteristics of trajectory data. The macro-scale features better represent the global trend, while the micro-scale ones determine the level of prediction accuracy. The T-CONV model is developed based on multi-layer convolutional neural networks to extract multi-scale features [6]. Their visualization of the latent layer reveals that the first 10% and the last 10% of a trajectory have a higher impact on the destination prediction. With such observation, they introduced local enhancement convolution (named T-CONV-LE model) to extract more detailed features in areas close to the start and end of the partial trajectory input. The authors subsequently introduced the multi-scope local enhancement model (T-CONV-LE-MUL) to further exploit the essential portion of a trajectory through a customized attention mechanism [16]. Given the prior knowledge of the importance of the start and end, an attention mechanism was introduced to define variable window sizes of local enhancement (instead of fixed 10%) around the start/end locations, and the attention weights were calculated explicitly based on historical traffic flow inside the local grid. This multi-scope local enhancement was found to further improve the prediction accuracy verified based on Porto and T-drive taxi datasets. However, the algorithm was subjected to several hyper-parameters: (i) the number of spatial grids; (ii) the levels of local enhancement; and (iii) the scaling factor of local enhancement. It is noted that the final destination location was predicted based on pre-clustered destination centers that were similar to [19]. The authors commented that predicting the destination based on these clustering points could help capture the semantics of each trajectory. However, there were no details on the setup of the pre-clustering algorithm and its generalization to new destinations or urban areas.

III. Preliminary

This section will introduce the common concepts and workflows of destination prediction based on vehicular trajectories.

A. Definition of Trajectory

A trajectory T is a sequence of spatial-temporal data points recording the movement of an object over time. Formally, a full trajectory is defined as,

$$T = [(\mathbf{s}_1, \mathbf{c}_1, t_1), ..., (\mathbf{s}_n, \mathbf{c}_n, t_n)], \tag{2}$$

where n is the length of the sequence, $\mathbf{s_t} = (x_t, y_t) \in \Re^2$ represents the (latitude, longitude) coordinates at time t, and the associated metadata $\mathbf{c}$ contains any available contextual information (e.g., vehicle ID, weather condition, etc.). In real-time applications, it is common that only the partially completed trajectory up to the k-th points, $\tau_1^k = [(\mathbf{s}_1, \mathbf{c}_1, t_1), ..., (\mathbf{s}_k, \mathbf{c}_k, t_k)]$, is available. The objective of a trajectory model (f) is to take the partial trajectory as the input and predict the target output (e.g., destination location).

B. Characteristics of Vehicular Trajectory

Trajectory data are available from many settings, such as walking, cycling, driving, trajectory by Point-of-Interests, etc. When utilizing a specific type of trajectory data, it is essential to understand its spatial, temporal, and contextual nature. Vehicular trajectory data usually cover part/whole of a city and span from minutes to a few hours. Besides, they faithfully follow the urban layout. As most urban road networks follow a radial pattern, GPS records show varying spatial densities in different parts of the city (i.e., multi-scale characteristics). A common observation is that a vehicle usually travels in the sequence of minor-to-major-to-minor roads. Thus, different portions contribute differently to the prediction (i.e., multi-scope characteristics). Most existing research efforts focus on improving the prediction accuracy by extracting more meaningful multi-scale and multi-scope features through deep learning models.

C. Spatial Discretization of Trajectory

Although it is feasible to feed GPS coordinates as direct inputs to the trajectory model, it is usually not adopted due to two reasons: (i) coordinates have limited representation of spatial connectivity; and (ii) trajectory records are commonly subjected to noises due to GPS drift or signal blockage. Therefore, raw trajectory data need to go through a pre-processing step: spatial discretization either by (i) road network or (ii) structured grids. In the former case, it utilizes the graph representation of the urban road links and converts a trajectory into a sequence of road nodes (analogous to words in a text sentence) through map-matching algorithms [22]. While in the latter case, it divides the two-dimensional urban space into structured grids (analogous to pixelated images) and presents a trajectory as a two-dimensional pattern or a sequence of grid IDs. In both cases, the spatial discretization process helps to reduce/remove the noises and embed spatial connectivity into the node/grid representations. The selection

of spatial discretization scheme is a critical design choice as it affects the subsequent learning techniques (e.g., sequential vs. convolutional) adopted to extract the multi-scale and multi-scope information.

IV. Methodology

This section introduces our proposed Location Recommendation System for Destination Prediction. When designing LRS4DP, we re-examined destination prediction by incorporating the domain knowledge learned from past studies and making trade-offs among design choices. Our driving principle is to improve the generalization ability by reducing empiricism and enhancing performance and scalability, which are achieved by formulating destination prediction as a Top-K location recommendation problem. Figure 1 illustrates the overall architecture of the LRS4DP framework.

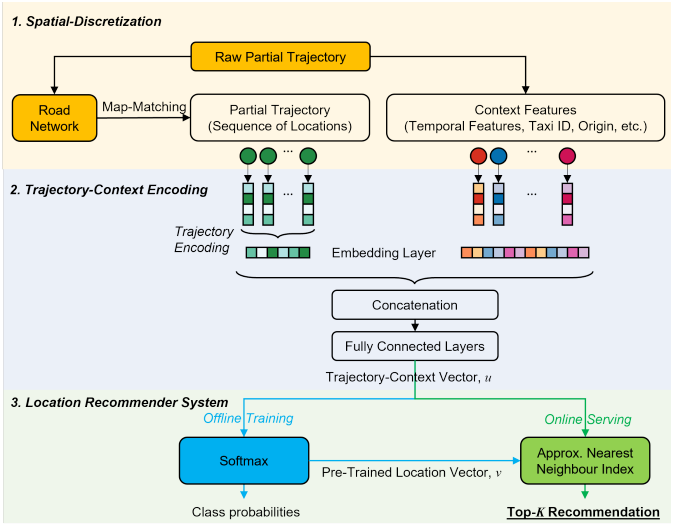


Fig. 1: Overview of Location Recommendation System for Destination Prediction

A. Spatial Discretization

The direct inputs to LRS4DP are the raw partial trajectories, and the first step is to map the trajectories into sequences of semantically meaningful spatial units. As introduced in Section III-C, the choice of spatial discretization scheme significantly influence the spatial precision and the subsequent learning techniques.

In terms of implementation, spatial discretization by structured grids is very straightforward. However, the grid resolution needs careful selection as details within a grid are lost. As a trajectory only passes a streak of grids, finer granularity leads to exponentially increasing grid numbers and the data sparsity issue. Therefore, it is necessary to adopt a multi-scale approach to capture a range of details and limit the computational expense through local refinement. The drawback is that it introduces a set of hyper-parameters that require tuning according to the urban layout. On the other hand, spatial discretization by road network results in sequences with better physical meanings as the road nodes are physical locations with topological constraints, and the density of nodes adapt automatically according to the road network density at different regions. Therefore, we promote spatial discretization by a road network as a more generic pre-processing step for vehicle trajectory data since it can better utilize the prior knowledge on the road network topology and does not require hyper-parameter tuning for capturing the multi-scale characteristics as for the grid-based scheme.

With the road network (G) as an additional input, we can convert the raw partial trajectories into sequences of road node IDs.

$$f_{map\_matching}\left(\tau_1^k, G\right) \rightarrow \tau_1^n \tag{3}$$

with,

$$\tau_1^k = [(x_1, y_1), ..., (x_k, y_k)] \tag{4}$$

$$\tau_1^n = [l_1, ...l_t, ...l_n] \tag{5}$$

where, τ_1^k is the sequence of raw coordinates with length k; τ_1^n is the map-matched sequence of location IDs with length n; $G = (V, E)$ is the road network represented by a list of nodes (V) and connecting edges (E); and $f_{map\_matching}$ is the map-matching function. Besides, we also prepare a list of m context features C for subsequent trajectory-context encoding.

$$C = \{c_1, ...c_j, ...c_m\} \tag{6}$$

B. Trajectory-Context Encoding

The second step is to encode the trajectory-context pair (τ_1^n and C) into a dense representation. As shown in Figure 1, each node ID is represented by a trainable latent vector $e_l \in \Re^{d_l}$ and the whole partial trajectory is represented by a sequence of embedding vectors,

$$\tau_1^n = [e_{l1}, ...e_{lt}, ...e_{ln}] \tag{7}$$

To further encode the trajectory into latent representation, we tested two encoding options with either (i) simple averaging: to understand the baseline performance without considering the sequential information.

$$emb_{traj} = \frac{\sum_i^n e_{lt}}{n} \tag{8}$$

or (ii) Gated Recurrent Unit (GRU): to understand the performance gained by adopting an RNN architecture.

$$emb_{traj} = GRU\left([e_{l1}, ...e_{lt}, ...e_{ln}]\right) \tag{9}$$

The context features (c_j) are also embedded into trainable latent vectors ($emb_{cj} \in \Re^{d_c}$). They are concatenated with the encoded partial trajectory vector (emb_{traj}), and then passed through one or multiple fully-connected layers to produce the final trajectory-context embedding vector $\mathbf{u}$.

$$u = W_u\langle emb_{traj}, emb_{c1}, ..., emb_{cm}\rangle + b_u \tag{10}$$

C. Location Recommender System

The third step is to train a classifier to predict the Top-K destinations. However, the number of candidate locations/nodes is in the order of $\mathcal{O}\left(10^4\right)$ to $\mathcal{O}\left(10^5\right)$ for an urban city, which adds significantly to the inference time due to heavy softmax calculations. Therefore, we propose to improve the efficiency by utilizing a recommender system framework. The central idea behind a recommender system is that, given an input query, it helps to quickly return the most relevant items out of a large pool of candidates. Such a system naturally fits our context of dealing with all the individual nodes of the entire road network at a city scale. By taking the partial trajectory and the associated context information as a proxy of the passenger/driver's interest, we propose to formulate destination prediction as a Top-K location recommendation problem that focuses on finding nodes that are likely to be around the actual destinations out of a location corpus.

1) Offline Training: During training, the trajectory-context vector is passed through a softmax layer to output the probability of each candidate location being the destination among the entire location corpus.

$$p_j = \frac{e^{\mathbf{v_i u}}}{\sum_{j \in V} e^{\mathbf{v_j u}}}, \tag{11}$$

where $\mathbf{u} \in \Re^D$ and $\mathbf{v_j} \in \Re^D$ are fixed-length (D) dense vectors representing the embedding of each trajectory-context pair and location, respectively. V is the size of the location corpus and commonly $\mathcal{O}\left(10^4\right)$ to $\mathcal{O}\left(10^5\right)$ for an urban city. These location embedding vectors ($\mathbf{v}$) are later extracted as the weights of the softmax layer and used to build an indexing structure for online serving.

2) Online Serving: During inference, to alleviate the extreme multi-class classification issue due to heavy softmax calculation, the trained classification model is only used to encode a trajectory-context pair into a dense vector $\mathbf{u}$. It retrieves the Top-K candidates by Approximate Nearest-Neighbor (ANN) Search with $\mathbf{u}$ as the query input. Given a candidate pool size of $\mathbf{V}$, replacing softmax with search methods will theoretically reduce the time complexity from $\mathcal{O}\left(V\right)$ to $\mathcal{O}\left(log(V)\right)$ [23]. In the present study, we utilize an open-source library FAISS[1] for index building and query using fast Hierarchical Navigable World graph (HNSW) method [23], [24]. We will test the actual run-time difference through experiments.

V. Experiments

This section introduces the details of the experimental design and numerical implementation.

A. Data Processing

1) Data Sources: We use two real-world taxi datasets to evaluate the proposed LRS framework for destination prediction. The first one is the Porto taxi dataset[2] widely used for destination prediction benchmarking [4], [6], [16], [19], [20].

[1]Version 1.7.1: https://github.com/facebookresearch/faiss

[2]Published in ECML/PKDD 15 Kaggle competition, https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

TABLE I: Comparison of Data Statistics

	Porto	Beijing
Spatial Coverage		
Latitude	41.052431 to 41.257678	39.756767 to 40.022693
Longitudinal	-8.456039 to -8.72791	116.201700 to 116.543634
Distance	22.6 km x 29.9km	29.3 km x 29.3km
Road Network		
Number of Road Nodes	72,435	24,616
Number of Road Links	173,048	55,882
Map-Matched Trajectories		
Number of Filtered Trajectories	1,333,609	3,309,348
Number of Unique Road IDs	27,848	23,615
Number of Unique Destination IDs	18,928	22,110
Matching Accuracy, d (Mean/Median)	0.074 / 0.059 km	0.131 / 0.092 km
Mean/Median Length of Road Links	0.104 / 0.071 km	0.189 / 0.130 km
Mean/Median Trajectory Length	47.4 / 43 nodes	27.9 / 24 nodes

It consists of 1.7 million trajectories generated by 442 taxis from Porto (Portugal) over a year (from 01 July 2013 to 30 June 2014). We use a second dataset from Beijing (China) to further verify the generalization ability of our model. It covers 3.3 million trajectories of 47,045 taxis collected within the Fifth Ring Road of Beijing over seven days (from 17 to 23 August 2015). Both datasets are in the format of sequences of timestamped geographical positions (longitude and latitude coordinates). We also include the commonly available metadata, such as the taxi ID. The general statistics are compared in Table I. Similar to previous studies [17], we remove trajectories that are too short (¡ 1 minute) or too long (¿ 2 hours) as these data are more likely affected by GPS errors or trip segmentation errors.

2) Spatial Discretization by Road Network: We implemented map-matching to align the trajectory trace to the road network through Leuven.MapMatching[3], an open-source toolbox that implemented the Hidden Markov Model (HMM) with non-emitting states method [22]. We downloaded the road network configurations from OpenStreetMap, and provided a comparison of the statistics between Porto and Beijing in Table I. Through map-matching, we convert the raw trajectories into sequences of node IDs, with the first/last node denoting the origin/destination node. For the partial trajectories, we use the median sequence lengths as general choices and add padding zeros when the trajectory is shorter. The map-matching accuracy is measured by the Haversine distance (d) between the map-matched destination node and the actual destination location, i.e., Equations 12 and 13 as below,

$$a = sin^2(\frac{\phi_2 - \phi_1}{2}) + cos(\phi_1)cos(\phi_2)sin^2(\frac{\lambda_2 - \lambda_1}{2}) \tag{12}$$

$$d = 2 \cdot r \cdot a \tan\left(\sqrt{\frac{a}{1-a}}\right) \tag{13}$$

where, ϕ is the latitude, λ is the longitude, d is the distance between two points, and r is the Earth's radius (e.g., 6371 kilometers). The final input features to the model include

- Context features: taxi ID, origin node ID, and temporal features (including quarter-hour, data of the week, week of the year based on the starting timestamp)
- Partial-trajectory in the form of a sequence of node IDs.
- Classification label: destination node ID

[3]Version 0.5.3: https://github.com/wannesm/LeuvenMapMatching

TABLE II: Benchmark Performance

Benchmarks	Discretization	Traj. Encoding	Formulation	MAE
MLP	GPS Points	MLP	Classification	2.81
Bi-RNN-W	GPS Points	RNN	Classification	2.60
T-CONV-Basic	Grids	CNN	Classification	2.70
T-CONV-LE	Grids	CNN	Classification	2.53
T-CONV-LE-MUL	Grids	CNN	Classification	2.44
ELM (Ensemble)	GPS Points	SVR+DBN	R+C	2.06
Trembr$_{f+b}$	Road Nodes	RNN	Regression	0.89*

B. Model Scenarios

As described in Section II, the existing models mainly differ in spatial discretization, trajectory encoding, and formulation of destination prediction. The baseline models are listed in Table II, which include

- MLP [19]: The winning solution of the Kaggle competition. A trajectory was represented by the first and last five GPS points and encoded by Multi-Layer Perceptron (MLP) with 500 neurons. They predicted the destination as the weighted averages of the pre-defined destination clusters;
- Bi-RNN-W [19]: Best model as claimed by the authors. It was mostly the same as MLP above, but the choice of trajectory encoder was bidirectional RNN with a 5-point window shift;
- T-CONV-Basic [6]: The trajectory was discretized by structured grids and encoded through a CNN model. The model formulated destination prediction as a classification and calculated the final coordinates as the weighted averages of the candidate grids;
- T-CONV-LE [6]: The improved version of T-CONV-basic with locally refined convolutional fields to explore finer details;
- T-CONV-LE-MUL [16]: A further improvement of T-CONV-LE by exploring the important regions through multi-scope local-enhancement areas automatically highlighted based on attention mechanism;
- ELM [20]: An ensemble method based on two models: Support Vector Regression (SVR) and Deep Belief Network (DBN). Both models take the first and last five GPS points as input and then predict the destination using direct regression by SVR or classification by DBN. The DBN model uses a 2-layer MLP with 500 neurons each;
- Trembr [17]: The trajectory is discretized into a sequence of road nodes and encoded by a bi-directional RNN model. The destinations are predicted as regression tasks with two linear models for latitude and longitude separately. However, the authors only trained and validated the model by assuming 80% of the full trajectory was known. Thus, the MAE appears much lower than the other models.

We note that for all the existing classification models, the candidate classes are the pre-defined destination clusters derived from historical destination locations using mean shift clustering [6], [16], [19], [20]. LRS4DP aims to achieve better precision and generalization by directly working on the individual road nodes and formulate destination prediction as a location recommendation problem. As illustrated in Figure 1, one main component of LRS4DP is the choice of trajectory encoder. In the present study, we aim to understand how its choice affects the expressiveness of the trajectory encoding and the final performance on destination prediction. We have tested both non-recurrent and recurrent trajectory encoders as below to understand their performance.

- LRS4DP + Averaging (LRS4DP-A): It averages the embedding vectors among all nodes without considering the order;
- LRS4DP + Gated Recurrent Unit (LRS4DP-GRU): It encodes the sequence of road node embedding through a recurrent neural network. GRU is one type of RNN model with only two gates (reset gate and update gate) to memorize the long and short-term memories.

Once the partial trajectory is encoded, it is concatenated with the context embedding and passed through the fully connected (FC) layers. To benchmark with [19] and explore the power of simple MLP architecture, we tested models LRS4DP-A1/2/3 with 1/2/3 FC layers (512, 512/256, 512/256/128 neurons), respectively. To quantify the additional performance margin by RNN-based encoder, LRS4DP-GRU is combined with 3 FC layers. Thus, the trajectory-context vector u and the location vector v both have a default dimension of 128 for LRS4DP-A3 and LRS4DP-GRU models.

C. Computational Details

1) Default Dimensions: As shown in Figure 1, all features are embedded into dense vectors. The location ID contains the highest cardinally (N=72,435/24,616 unique node IDs for Porto/Beijing). Experiments show that increasing the dimension of location embedding leads to better model accuracy but longer training time. We adopt a dimension of 16 (approximately log_N as a general choice) when comparing against the state-of-the-art performance. The embedding dimensions of other categorical features are all set to 8. The hidden state size of the GRU layer is set to 16. Each of the fully connected layers is followed by a ReLU activation. The dropout rate is set to 0.2.

2) Training Configuration: Instead of randomly splitting the data as in previous studies [19], we sort the trajectory by time and then take the earliest 90% as training data, followed by 5% validation data and the latest 5% as the test dataset. The maximum training cycle is set to 20 epochs with early stopping, which is found sufficient for model convergence. Categorical cross-entropy is adopted as the loss function for the classification model. Through preliminary study, we found that it produced better performance when adopting SGD with Adam optimizer (momentum=0.9), and the learning rate is set to 0.01 for the Porto dataset and 0.1 for the Beijing dataset.

3) Evaluation Metrics: Similar to previous studies, the final accuracy of destination prediction is measured by the Mean Absolute Error (MAE) [19],

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(d_i) \tag{14}$$

TABLE III: Performance of LRS4DP: Porto (based on Top-50)

Model	MAE (km)		Coverage (%)	
	Softmax	ANN	Softmax	ANN
LRS4DP-A1	2.24	3.74	27.5	11.1
LRS4DP-A2	2.06	2.26	29.9	27.1
LRS4DP-A3	1.86	2.09	35.5	30.0
LRS4DP-GRU	1.73	1.74	38.8	37.3

where d_i is the Haversine distance between the predicted and actual destination coordinates, and n is the number of data instances. Besides, we also analyze the coverage and run-time. Given the proposed Top-K formulation, the predicted final destination coordinates are calculated as the weighted averages of the Top-K candidate locations.

VI. Results

This section compares the performance of destination prediction based on the Top-K recommendation against the state-of-the-art algorithms for the Porto taxi dataset and assesses the trade-off between speed and run time. We will then validate the generalization ability of our proposed LRS4DP framework by assessing the trends in the Beijing taxi dataset and perform sensitivity analysis on the K-value to guide engineering applications. Besides, we also apply LRS4DP framework to other prediction tasks and examine the location embedding semantics, which demonstrates the capability of LRS4DP for supporting various downstream transportation and location-based service applications.

A. Destination Prediction

Table III summarizes the performance of LRS4DP models (MAE and coverage) for Porto taxi datasets for both offline training and online serving scenarios, which are labeled as "Softmax" and "ANN" according to the method of retrieving the Top-K candidates. We use $K = 50$ to demonstrate the results.

1) Offline Prediction by Softmax: We first examine the performance of offline Top-K destination prediction through Softmax, which outputs the probability of each candidate road node being the destination. Comparing the MAE under the "Softmax" column in Table III with benchmark models in Table II, we found that even a simple model (LRS4DPA1) can achieve better accuracy (2.24 km) than those benchmarks built upon RNN and CNN architectures (2.60km for Bi-RNN-W and 2.44km for T-CONV-LE-MUL). It proves the advantage in spatial precision by formulating the prediction on individual road nodes. Increasing the number of FC layers further improves the accuracy to 2.06 km for LRS4DPA2 with FC512/256 and 1.86 km for LRS4DPA3. Preliminary studies showed that further increasing the FC layers brought minor improvement. The incorporation of recurrent architecture (GRU) provides further improvement (1.73 km) which highlights the importance of sequential information for trajectory modeling.

As shown in the overall LRS4DP architecture (Figure 1), the weights of the software layer are extracted as the embedding vectors for all the candidate nodes. The ranking and Top-K prediction based on softmax calculation is essentially the same as the exact search method with similarity represented by the dot products between trajectory-context vector u and pre-trained location vector v. Therefore, offline prediction by softmax sets the upper bound accuracy for the online prediction based on ANN search to be discussed next.

2) Online Prediction by ANN Search: To improve scalability, we adopt a recommender system framework and retrieve the Top-K candidates through an online matching process. As described in Section IV-C, given a query vector u during inference, the recommender system matches the Top-K candidate locations through ANN search by utilizing the location embedding vectors (i.e., weights of the softmax layer) trained offline. It avoids the heavy softmax calculation associated with extreme multi-class classification, thus improving the run-time scalability. As shown in Table III, the performance of ANN deviates lesser from Softmax for more complex trajectory encoders, suggesting that the location embedding is trained semantically better with more sophisticated architecture.

3) Effect of Trip Percentages: For partial trajectories closer to the completion of the trips, we shall expect higher accuracy in terms of destination predictions. To give the readers better anticipation of the level of variation and to allow the benchmarking with Trembr$_{f+b}$ model trained and tested under the condition that 80% of the full trajectory was known [17], we present the MAE over a full spectrum of trip completion percentages in Figure 2. The completion percentages are calculated based on the number of nodes rather than actual travel distances. As described in Section V-B, Trembr$_{f+b}$ achieved an MAE of 0.89 km by formulating destination prediction as a regression task with RNN and road node embedding. The major differences of our proposed LRS4DP framework are (i) we formulate destination prediction as a Top-K prediction for better capturing the location semantics and facilitating downstream tasks; and (ii) we designed an end-to-end model to train location embedding and trajectory embedding simultaneously, while Trembr$_{f+b}$ trained them separately. As shown in Figure 2, Trembr$_{f+b}$ is marked using a red cross marker, which lies between the results of LRS4DP-A3 and LRS4DP-GRU models. It signifies the importances of RNN model architecture for trajectory modeling. Besides, LRS4DP gives better accuracy by formulating destination prediction as a classification formulation. Trembr$_{f+b}$ also employed a customized loss function, while LRS4DP framework is more generic for its model architecture and implementation. It is also worth noting that the trip completion percentages are practically unknown in real-time applications.

B. Trade-off between Speed and Accuracy

Comparing the offline and online performance, it is clear that the online ANN search will trade accuracy for run-time speed. To explore the trade-off, we compared the performance of softmax calculation with ANN search using the same GRU model under different settings. Here, we focus on two key ANN search parameters: M and e_f which influence the index-

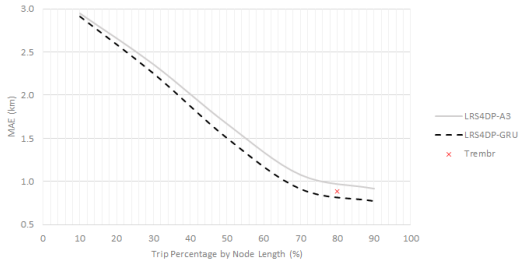


Fig. 2: Effect of trip completion percentages on accuracy (Porto)

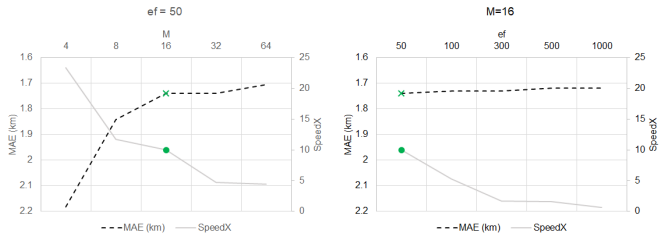


Fig. 3: Trade-off between accuracy and speed by varying M and e_f based on Porto dataset. Left: M = 4 to 64 with e_f = 50; Right: e_f = 50 to 1000 with M = 16. MAE and SpeedX values from the common test case with M=16 and e_f=50 are marked with green crosses and dots, respectively.

construction and search process, respectively. M represents the number of bi-directional links created for every new element, and a higher M-value is more suitable for higher-dimensional datasets but increases the memory requirement and search effort; e_f is the size of the dynamic list for the nearest neighbors used during the search, and a higher e_f leads to more accurate but slower search.

As shown in the first plot of Figure 3, we adopt different indexing complexities by varying M from 4 to 64 and fixing e_f at 50 (i.e., same as K value). Speed multiplier (SpeedX) is calculated by dividing the run-time of Softmax calculation by that of ANN search. It shows that the accuracy of destination prediction improves given more search effort (lower MAE with higher M), and the run-time advantage reduces (lower SpeedX with higher M). Interestingly, we can achieve 4.4 times speed improvement without losing accuracy for the Porto dataset. We then fix M at 16 and test the sensitivity to search effort by varying e_f from 50 to 1,000. As shown in the second plot in Figure 3, the accuracy of ANN search slightly improves with larger e_f. However, the search time drops quickly and becomes inefficient (SpeedX ¡ 1.0 with e_f=1,000). In the present study, we found that M=16 and e_f=50 work well for the Top-50 prediction for both Porto and Beijing datasets. We will further discuss the sensitivity in the following section on generalization.

C. Generalization

Assessments on the Beijing dataset further confirm the trends we observed over the Porto dataset. As shown in Table IV, the performance improves with more complex architecture, and the GRU model gives the best results. We also observe that LRS4DP-A3 achieves a similar MAE (2.80 km) as LRS4DP-GRU (2.79 km), while LRS4DP-GRU provides a higher coverage. It suggests that simple MLP architecture yields promising results for Beijing data. It could be due to Beijing's relatively larger spatial extent and coarser road network. Besides, the development of Beijing's Fifth Ring region is relatively more uniform (refer to Figure 4). Therefore, the sequential information is less influential compared the Porto dataset. In general, GRU appears to be a good starting point for practitioners when investigating a new area of interest.

TABLE IV: Performance of LRS4DP: Beijing(Top-50)

Model	MAE (km)		Coverage (%)	
	Softmax	ANN	Softmax	ANN
LRS4DP-A1	3.02	4.10	20.5	14.4
LRS4DP-A2	2.93	3.09	27.3	26.2
LRS4DP-A3	2.75	2.80	33.5	31.2
LRS4DP-GRU	2.65	2.79	37.3	33.5

Further comparison between Table III and Table IV shows that the MAE is higher for the Beijing dataset (e.g., 2.79 km or 1.6 times the 1.74 km for Porto under the same GRU model). It is critical to understand if such difference is due to poor generalization or the fundamental data distribution. By examining the spatial scales of the cities and their road networks, we believe this is mainly due to the nature of data. As shown in Figure 4, the road network of Porto expands radially from the city center and irregularly in the suburban areas, while that of Beijing follows a more structured and nested configuration. To quantify the difference, we compare the statistics in Table I, the median length of road links in Beijing (0.130 km) is 1.8 times the value of Porto (0.071 km), and the median ground truth error (0.092 km), measured between destination node labels and actual destination locations, is 1.6 times the value of Porto (0.059 km). Therefore, by applying the same model architecture and hyper-parameters, we are able to achieve a comparable level of accuracy among cities with different scales and layouts. The performance of LRS4DP is expected to adapt automatically according to the local road network resolution. Besides, by defining the candidate class pool according to the entire road network, the appearance of new destinations does not require model re-training comparing to existing models based on pre-clustered destination centers [6], [16], [19], [20].

Fig. 4: Comparison of traffic network. Left: Porto; Right: Beijing

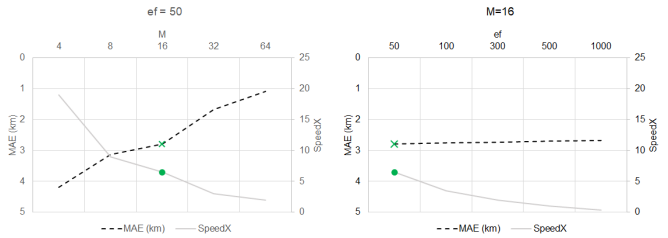


Fig. 5: Trade-off between accuracy and speed by varying M and e_f based on Beijing dataset. Left: M = 4 to 64 with e_f = 50; Right: e_f = 50 to 1000 with M = 16. MAE and SpeedX values from the common test case with M=16 and e_f=50 are marked with green crosses and dots, respectively.

We also analyzed the trade-off between accuracy and speed for the Beijing dataset (Figure 5) and observed similar trends as in Figure 3. The speed advantage is more significant for Porto dataset than the Beijing dataset due to the larger location corpus (72.4k nodes vs. 24.6k nodes). For real applications, it is possible to further reduce the inference time by adopting smaller M and e_f (with some accuracy trade-off).

D. Sensitivity to K-Value

Figure 6 shows the performance of LRS4DP-A3 and LRS4DP-GRU under Top-K ANN search ranging from K=1 to 1000. The accuracy generally improves when K increases from 1, reaches a plateau around K = 30 to 100, and starts to degrade with higher K values. K=50 appears a good starting point that provides stable results for both Porto and Beijing datasets. We also noticed that for the Porto dataset, even the MAE of Top-1 prediction for LRS4DP-A3 (2.44 km) is still better than most of the baseline models (Table II). It again demonstrates the advantage in spatial precision of our problem formulation as classification based on individual road nodes.

E. Location Embedding Semantics and Other Applications

Previous studies with representation learning demonstrated that the location embedding automatically captures the topological structure of the road network. Locations that are physically connected will appear closer to each other when visualized in the latent vector space [14]. Therefore, it is interesting to explore further the semantics learned by the location embedding and if there can be any generalization among embedding trained on different geospatial tasks.

To explore such semantics, we trained separate models using the same LRS4DP framework with different labels for two additional tasks: predicting (i) the immediate next location and (ii) the passing-by location by remaining future trajectory. The model architecture and data remain the same as those for destination prediction, except replacing the labels. For passing-by location prediction, the data label is a location randomly sampled from the remaining future trajectory. The embeddings are then extracted from the three models and used to recall the Top-K locations. The comparison of coverage for different tasks is presented in Figure 7. Each sub-figure represents

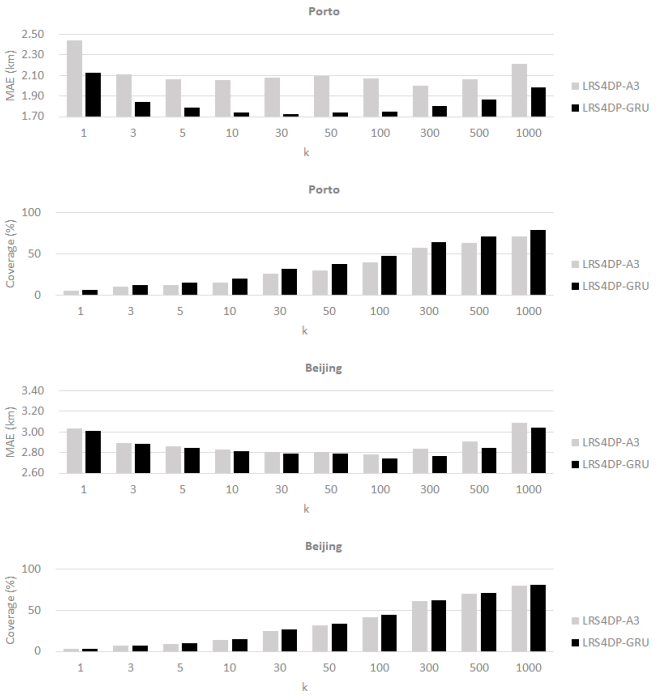


Fig. 6: Sensitivity to K value. Top to Bottom: Porto-MAE; Porto-Coverage; Beijing-MAE; Beijing-Coverage.

the coverage of three types of labels under a single type of embedding. It is obvious that the task-specific embedding performs the best. However, the locations recalled back by the next location and destination based embeddings indeed show a certain degree of overlap. To visualize the distribution of locations recalled by different embeddings, the Top-50 recalled locations based on the same single partial trajectory query are plotted on GoogleMap in Figure 8.

This demonstrates that LRS4DP framework can be generically applied to various prediction tasks and support multiple downstream transportation and location-based service applications. For example, information about possible destinations might be aggregated and useful for predicting the spatial distribution of taxi supply. Such information is useful in real-time applications, such as dynamic pricing. Besides, the next location and passing-by location predictions can be utilized to support last-mile logistic services with stochastic pickup and delivery orders and dynamic routing.

VII. Conclusions

The present study works on destination prediction based on partial trajectories as a fundamental and widely applicable problem in urban mobility. The practical implementation of existing deep learning-based destination prediction models is not trivial due to generalization and scalability issues. We developed a general solution framework, LRS4DP, by integrating several technologies (map-matching, deep learning, and recommender system). By adopting a node-based spatial discretization scheme through map-matching, LRS4DP can

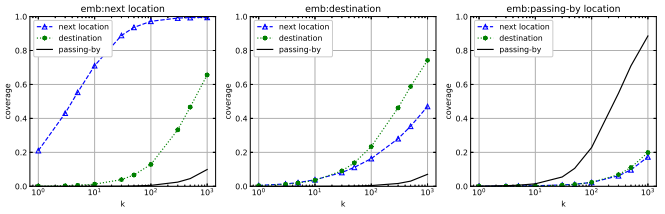
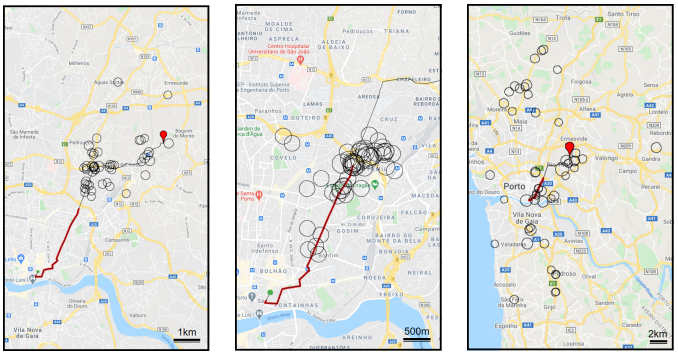


Fig. 7: Coverage of recall subjected to various embedding semantics and tasks (a) destination (b) next location, and (c) passing-by location



(a) (b) (c)

Fig. 8: Visualization of Top-50 locations recalled for (a) destinations, (b) next locations and (c) passing-by locations

adapt according to the local road network density and generalize to different urban layouts. Comparing to the state-of-the-art algorithms, LRS4DP's Top-K formulation based on individual road nodes leads to fundamentally better spatial precision and prediction accuracy even with simple model architectures. LRS4DP's scalability is inherited from the recommender system approach by ANN search with the flexibility to trade-off between accuracy and run time. The evaluation based on two real-world taxi datasets demonstrates the significant performance improvement and the robustness of LRS4DP to different urban scales and layouts. The LRS4DP framework is also generically applicable for other prediction tasks (e.g., next location and passing-by location predictions) and thus capable to support various downstream transportation and location-based service applications. In future studies, it will be useful to further improve the performance by exploring more sophisticated recommendation algorithms and develop a common system to support various downstream applications.

REFERENCES

[1] J. Zhang, H. Che, F. Chen, W. Ma, and Z. He, "Short-term origin-destination demand prediction in urban rail transit systems: A channel-wise attentive split-convolutional neural network method," *Transportation Research Part C: Emerging Technologies*, vol. 124, p. 102928, 2021.

[2] S. E. Huang, Y. Feng, and H. X. Liu, "A data-driven method for falsified vehicle trajectory identification by anomaly detection," *Transportation research part C: emerging technologies*, vol. 128, p. 103196, 2021.

[3] C. Zhang, J. Bin, W. Wang, X. Peng, R. Wang, R. Halldearn, and Z. Liu, "Ais data driven general vessel destination prediction: A random forest based approach," *Transportation Research Part C: Emerging Technologies*, vol. 118, p. 102729, 2020.

[4] P. C. Besse, B. Guillouet, J.-M. Loubes, and F. Royer, "Destination prediction by trajectory distribution-based model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2470–2481, 2017.

[5] Y. Endo, K. Nishida, H. Toda, and H. Sawada, "Predicting destinations from partial trajectories using recurrent neural network," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2017, pp. 160–172.

[6] J. Lv, Q. Li, Q. Sun, and X. Wang, "T-conv: A convolutional neural network for multi-scale taxi trajectory prediction," in *2018 IEEE international conference on big data and smart computing (bigcomp)*. IEEE, 2018, pp. 82–89.

[7] X. Kong, Z. Xu, G. Shen, J. Wang, Q. Yang, and B. Zhang, "Urban traffic congestion estimation and prediction based on floating car trajectory data," *Future Generation Computer Systems*, vol. 61, pp. 97–107, 2016.

[8] J. Sun and J. Kim, "Joint prediction of next location and travel time from urban vehicle trajectories using long short-term memory neural networks," *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103114, 2021.

[9] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut, "A taxi dispatch system based on prediction of demand and destination," *Journal of Parallel and Distributed Computing*, vol. 157, pp. 269–279, 2021.

[10] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye, "A taxi order dispatch model based on combinatorial optimization," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 2151–2159.

[11] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility markov chains," in *Proceedings of the first workshop on measurement, privacy, and mobility*, 2012, pp. 1–6.

[12] D. Xie, F. Li, and J. M. Phillips, "Distributed trajectory similarity search," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1478–1489, 2017.

[13] F. Valdés and R. H. Güting, "A framework for efficient multi-attribute movement data analysis," *The VLDB Journal*, vol. 28, no. 4, pp. 427–449, 2019.

[14] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling trajectories with recurrent neural networks." IJCAI, 2017.

[15] J. Xu, J. Zhao, R. Zhou, C. Liu, P. Zhao, and L. Zhao, "Destination prediction a deep learning based approach," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[16] J. Lv, Q. Sun, Q. Li, and L. Moreira-Matias, "Multi-scale and multi-scope convolutional neural networks for destination prediction of trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3184–3195, 2019.

[17] T.-Y. Fu and W.-C. Lee, "Trembr: Exploring road networks for trajectory representation learning," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 1, pp. 1–25, 2020.

[18] Z. Gui, Y. Sun, L. Yang, D. Peng, F. Li, H. Wu, C. Guo, W. Guo, and J. Gong, "Lsi-lstm: An attention-aware lstm for real-time driving destination prediction by considering location semantics and location importance of trajectory points," *Neurocomputing*, vol. 440, pp. 72–88, 2021.

[19] A. De Brébisson, É. Simon, A. Auvolat, P. Vincent, and Y. Bengio, "Artificial neural networks applied to taxi destination prediction," *arXiv preprint arXiv:1508.00021*, 2015.

[20] X. Zhang, Z. Zhao, Y. Zheng, and J. Li, "Prediction of taxi destinations using a novel data embedding method and ensemble learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 68–78, 2019.

[21] J. Zhao, J. Xu, R. Zhou, P. Zhao, C. Liu, and F. Zhu, "On prediction of user destination by sub-trajectory understanding: A deep learning based approach," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1413–1422.

[22] W. Meert and M. Verbeke, "Hmm with non-emitting states for map matching," in *European Conference on Data Analysis (ECDA), Date: 2018/07/04-2018/07/06, Location: Paderborn, Germany*, 2018.

[23] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[24] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *arXiv preprint arXiv:1702.08734*, 2017.