東京大学大学院新領域創成科学研究科
社会文化環境学専攻

2023年度
修　士　論　文

Recommender System Based Trajectory Prediction
推薦システムに基づく軌跡の予測

2023年7月14日 提出
指導教員　宋　　軒　准教授
小林　博樹　　教授

莫　宇
MO, YU

Abstract

In this research, we propose a novel method, titled Recommender System based Trajectory Prediction (RSTP), for predicting grid-based trajectories. This method synergizes the principles of the recommender system with trajectory prediction, encapsulating both spatiotemporal and attribute features. This novel approach capitalizes on the prowess of Long Short-Term Memory (LSTM) networks and DeepWalk, a prevalent graph embedding technique, to extract high-level features from both temporal and spatial domains. We further incorporate an activation unit, borrowed from Deep Interest Network (DIN) models, to model interactions between temporal and spatial features before input into a Multi-Layer Perceptron (MLP). The proposed approach balances the trade-off between prediction accuracy and the scope of the research area. Through extensive experiments, RSTP has demonstrated substantial potential in handling the complexity of trajectory prediction tasks. In some instances, the proposed method exhibits state-of-the-art performance, particularly when larger and more diverse datasets are employed. Our findings suggest that the application of recommender system principles could herald a new direction for future trajectory prediction research.

Key Word: Recommender System, Trajectory Prediction, Spatiotemporal Data modeling, Urban Planning, Congestion Control.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Trajectory prediction, a cornerstone technology in the development of intelligent transportation systems, plays a substantial role in the evolution of smart cities. This technology not only tackles Socio-Cultural and Environmental issues, but also elevates daily life by enhancing services such as Mobility as a Service (MaaS). MaaS, an integrated digital platform, consolidates multiple transport services into one on-demand service, offering a comprehensive and efficient alternative to private vehicle ownership.

For example, predicting future locations from a given trajectory, defined by a sequence of location points like longitude and latitude pairs, is a crucial aspect of traffic network optimization, congestion control, and route planning. With accurate predictions, decision-makers can develop improved traffic road designs that minimize congestion and energy waste, ultimately fostering an environmentally friendly society.

As integral components of MaaS, smartphone applications like Uber (an American company that offers a ride-hailing service, connecting drivers and passengers via a mobile app for convenient, on-demand transportation as an alternative to traditional taxi services), Google Map and Apple Map, (digital mapping services provided by Google and Apple respectively), have recently begun to provide services that suggest potential next locations based on the user's current situation and historical information. For instance, when you finish work at 6 pm and open these applications, they might recommend directions to a restaurant or home, informed by your past trajectory information. These features, driven by advancements in trajectory prediction technologies, significantly enhance the user experience and facilitate daily life.

1.2 Purpose & Contributions

Despite extensive research has been conducted in the field of trajectory prediction, providing accurate predictions of the next location based on past trajectories remains a challenging problem, influenced by the following aspects.

(1) Individual vs. Collective: Currently, trajectory prediction research predominantly falls into two main categories: Individual-based trajectory prediction and Collective-based trajectory prediction. The primary distinction between these two categories lies in their research scope and subjects. More specifically, the individual-based trajectory prediction approach focuses on a specific user, and usually limited to a relatively small area. For instance, a pedestrian on a sidewalk segment[1, 2] or a car on a highway stretch[3, 4]. While individual-based approaches often exhibit high accuracy within their limited area of focus, their

performance significantly decreases when applied to larger areas, sometimes even failing to provide reliable predictions. Few studies have explored individual vehicle trajectory prediction over cities and oceans[5], but they still face challenges with accuracy in such scenarios.

Conversely, the focus of collective-based trajectory prediction shifts from individual to group. As the trajectories of groups cannot be easily represented by a sequence of location points, the variation in density within a specific area is considered instead. For instance, predicting citywide crowd dynamics when big events happen[6], predicting a large-scale citywide crow density and flow in the daily life through the trajectory data[7], etc. It is important to note that while collective-based approaches do not directly predict individual trajectories, the underlying data used, such as population density, car flow, and crowd flow, are derived from trajectory data. Typically, these collective-based approaches analyze data on a large-scale, encompassing entire cities, to offer valuable insights for areas like traffic management and urban computing. However, due to their emphasis on collective behaviors, these approaches are not designed to provide accurate individual location/trajectory predictions. Their primary focus lies in analyzing and understanding group movement patterns rather than predicting individual locations in the next moment.

(2) Various complex factors: The prediction of the next location based on given trajectories is influenced by a multitude of complex factors. These include temporal correlations, spatial dependencies, the preferences or habits of the trajectory owner, as well as external factors like weather conditions and the day of the week. The significance of spatiotemporal correlations and dependencies in trajectory prediction problems has been demonstrated[8]. Moreover, these factors can

interact with each other in real-world scenarios. For instance, commuters may alter their commuting routes based on weather conditions. On a rainy day, they might choose to take a bus to work and prioritize a quick return home, while on a sunny day, they may opt to ride a bike to work and socialize at a restaurant with friends until late at night. To improve prediction performance, it is crucial to implicitly consider these factors within the prediction method. Additionally, as there are various transportation modes available, predicting trajectories involving multiple modes becomes more intricate.

To sum up, accurately predict the individuals' next locations through trajectory prediction approach is a challenging problem, especially when the area becomes large, like in the whole city. To address these challenges, this paper introduces a novel approach for city-wide multi-modal trajectory prediction, termed as (RSTP). This approach deviates from conventional trajectory prediction as it employs a grid-based trajectory instead of the original GPS point sequences. RSTP harnesses the strengths of recommender systems and grid representation, and it strives to achieve a balance between prediction accuracy and applicability across a broad geographic scale. Fig.1.1 is a grid-based trajectory demo of a arbitrary trajectory in United State. The primary contributions of this thesis can be summarized as follows:

- Proposed a novel trajectory prediction method, Recommender System based Trajectory Prediction (RSTP), which is based on deep recommendation systems.
- Proposed a spatiotemporal framework to learn the spatial and temporal correlations and dependencies from the raw GPS sequence in RSTP. In detail, a) Recurrent Neural Network (LSTM) that learn the temporal dependencies of the obtained feature map grids and embeddings from external factors; b)

Utilize DeepWalk to model the spatial information from graph derived from the raw GPS data; c) Apply activation unit to model complicate correlations and dependencies cross the temporal dimension and spatial dimension,

- Conduct extensive experiments on two large real-world datasets which consists of GPS points generated by mobile phone users in Tokyo and Osaka. Extensive experiments were conducted to demonstrate the feasibility of proposed model in the trajectory prediction tasks.

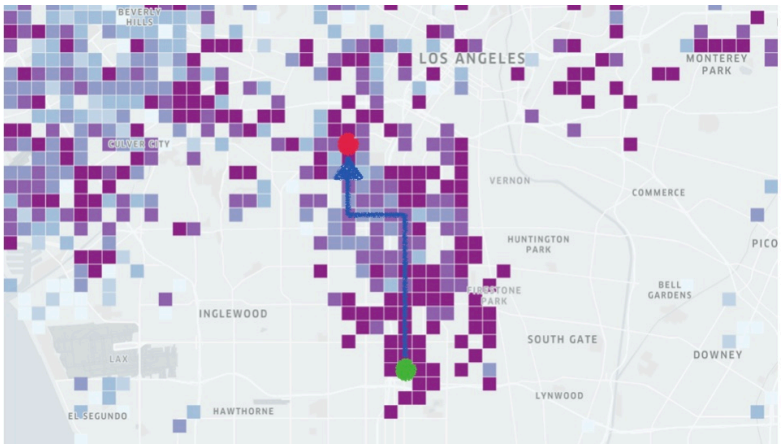


Fig. 1.1: The green dot is the starting point and the red point is the destination. The blue line is the actual grid-based trajectory. Each colored grid represents an area that has been traversed by individuals, with the depth of the color signifying the number of people that passed through that area within a certain time frame. The deeper the color, the more visiting trajectories in the area.

1.3 Structure

This thesis consists of 6 chapters in total, including the this *Introduction* Chapter,

- **Chapter 1**, gives the introduction of the problem that this paper focused on, mainly including the background, purpose and contributions.

- **Chapter 2**, delivers a literature review of the related work.
- **Chapter 3**, introduces preliminary knowledge and definition of the problem this paper focused on.
- **Chapter 4**, explain the details of experiments.
- **Chapter 5**, draw a conclusion and gives the future work for the further study.

Chapter 2

Related Work

Ever since the inception of the urban computing concept[9], the Trajectory Prediction has emerged as a vibrant area of research. Based on application scenarios, the research can be broadly divided into two main categories, i.e., individual-based trajectory prediction and collective-based trajectory prediction. In the rest of this section, I will give a brief literature review about those two type of approaches. Limited to the length of the paper, I only mention a few closely related researches.

2.1 Individual-based Trajectory Prediction

Individual-based trajectory prediction focuses on forecasting the future movements of a single entity, such as a person or vehicle, based on its historical trajectory data. This type of prediction has wide-ranging applications from navigation services, traffic management, to social network analysis. Depending on the data employed, this methodology can be bifurcated into 2 categories, i.e., Traditional Trajectory based Approaches and Grid Trajectory based Approaches.

2.1.1 Traditional Trajectory based Approaches

In this paper, traditional trajectory means the trajectory represented by GPS points. Traditional trajectory based on trajectory prediction has been studied extensively since several years ago (Alanhi 2016[10], Ma 2019[11], and Huang 2019[2]). At first, the basic idea of this type of approach is to use time series analysis techniques, such as Autoregressive Integrated Moving Average model (ARIMA), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) and so on. For example, Chen et al[1] shows a novel approach to use a modified LSTM to predict the pedestrian trajectory with complicated human-to-human interactions. And Yang et al[12] uses a extended ARIMA model called Traj-ARIMA to predict the cars' speed, and this method can be also applied into trajectory data prediction. As research progresses, in addition to modeling temporal information, an increasing number of researchers[2][13][14] are discovering that modeling spatial information is also crucial for studying trajectory data, which involves both time and space dimensions. Xu et al[15] proposed a novel Spatial-Temporal Attentive Neural Network called Tra2Tra to capture the complex spatial-temporal feature, which is based on the previous researches[1, 10]. Their most important contribution is to consider the spatial features into a temporal model. Varshneya et al[16] also pointed out that the spatial information matters in the dynamic human trajectory prediction task. In conclusion, both temporal and spatial information significantly influence the effectiveness of trajectory prediction tasks. Current research primarily focuses on predicting near-future locations, such as those in the next 10 seconds, 1 minute, or 3 minutes. These studies aim to maximize accuracy within a confined area due to application scenarios that demand high precision, such as autonomous car navigation, pedestrian behavior detection, etc. However, besides accuracy, data scale is a crucial factor in other contexts like urban computing and road network planning. Consequently, the accurate prediction of city-wide

trajectories represents a formidable challenge.

2.1.2 Grid with Trajectory Approaches

To address the problem outlined in the previous subsection, namely, the inability to accurately predict trajectories across an entire city, an increasing number of techniques, such as grid-based trajectory representations, are garnering attention in recent trajectory prediction research. Works by [17, 18, 19] have recognized that, with the assistance of a grid, the performance of trajectory prediction can be improved in various scenarios, including pedestrian trajectory prediction and autonomous vehicle navigation systems. Kim et al.[4] demonstrated that grid-based trajectory prediction is more efficient than traditional trajectory prediction methods when utilizing the same LSTM models.Their experimental results showed that predictions based on grid trajectories performed better than traditional ones. While Kim's work primarily focuses on enhancing prediction performance in localized areas, given these benefits, we can hypothesize that employing grid-based trajectory techniques could be instrumental in addressing the challenges of accurately predicting city-wide trajectories. Incorporating this technique into their research, Zhang et al.[20] successfully developed a Sequence-to-Sequence Model for worldwide vessel trajectory prediction. In summary, with grid techniques, the trajectory prediction performance can be further improved in many cases. And in some cases like crowd prediction/control, traffic management and etc., grid techniques may be one of the key techniques to solve the inability to accurately predict city-wide even world-wide trajectories.

2.2 Collective-based Trajectory Prediction

Contrary to the trajectories in individual-based trajectory prediction tasks, defining the trajectory in collective case is challenging , as different individuals within a group may have varying trajectories. Furthermore, in most use cases, the focus leans towards the alterations in collective properties, such as crowd density, vehicle flow, average speed, and the like, rather than changes in individual trajectories. However, it's important to note that all this data originally stems from individual trajectory data. This is the reason why I refer to this brunch of approaches as Collective-based Trajectory Prediction. Additionally, grid techniques, which is key part of the method this thesis proposed, is widely applied in this approaches, thus it is necessary to introduce related researches here.

Zhang et al.[21] developed a novel system based on Convolutional Neural Network (CNN) to predict the citywide crowd flow. In their work, they partitioned the city into a $I \times J$ grids and then mapped each GPS coordinate into a grid cell for the first time. After that, more and more researches[22, 23, 24] begins to adopt the grid techniques in the collective-based trajectory prediction task, or crowd flow prediction task. With the integration of the grid concept, a completely distinct approach employing Convolutional Neural Networks (CNN) has emerged.

2.2.1 Sequential Model based Approaches

Given that sequential models are designed to process time-series (sequential) data such as trajectory data and flow data, it's intuitive to apply these models to collective-based trajectory prediction tasks. The effectiveness of sequential models, including GRU, LSTM, and others, has been demonstrated in recent research[25, 26,

27, 28]. As research advances, it has been discovered that spatial information is equally important as temporal information in the trajectory prediction task. Therefore, considering the spatial features in sequential models has become an increasingly popular research topic, building on previous studies. Tang et al.[29] proposed a novel deep learning model, namely spatial–temporal recurrent neural network, to predict crowd flows, which merges temporal features of crowd flows and spatial features of roads simultaneously. As previously mentioned, a variety of factors such as transportation modes can also impact prediction outcomes, hence, Zhou et al.[30] proposed a relation-specific transformation model to deal with these heterogeneous relations. Based on the processed heterogeneous information, a modified LSTM model is used to finally predict the sequential crowd flow and demonstrate the superiority of of the proposed model. Additionally, the fusion of convolutional networks and sequential networks demonstrated its effectiveness in the work of Ali et al.[31]. In this study, they employed a neural network known as ConvLSTM to model spatiotemporal information and intricate correlations. Furthermore, other factors such as weather and day of the week were taken into consideration through an attention mechanism. Also, there are some studies which using the traditional technique like spatial clustering to model spatial information. For example, Fan et al.[32] developed a predicting-by-clustering framework to address the problem that unexpected change in crowd flow when emergent situations or yearly events, which provide a new idea to handle those irregular cases. Through these studies, we can see that not only does the modeling of spatiotemporal features impact the final prediction results, but also how external factors like weather and unusual activities is critically important to trajectory prediction research.

2.2.2 Convolutional Model based Approaches

With the introduction of the grid concept into trajectory prediction, CNN models, which are typically unrelated to trajectory prediction, can now be applied to the task. The basic idea is to treat each grid as a pixel and view a trajectory or an entire map as an image. Spatial features can then be extracted through the CNN model. More and more studies show the power of CNN model in grid-based trajectory prediction task[33, 34, 35, 36]. Given that this thesis does not extensively utilize CNN models, I will not go into great detail about methods of trajectory prediction using CNN models.

In summary, while current research on trajectory prediction is quite extensive, it typically either focuses on precise predictions for individual trajectories within a small scope, or studies overall movement trends within a larger scale, such as city-wide[1] even world-wide[16]. Therefore, I propose an innovative trajectory prediction model based on recommender systems in this thesis. This model effectively addresses the aforementioned problems, achieving a good balance between accuracy and scope. The proposed model are majorly refer to the previous researches in recommender systems[37, 38] and trajectory prediction[21, 33].

Chapter 3

Preliminary

In this section, I first present several preliminaries and give the formal definition of our problem.

3.1 Trajectory

3.1.1 Grid Map

Numerous interpretations of a location exist, varying in granularity and semantic implications. In this study, instead of representing a location with a specific longitude and latitude coordinate pair in most of the digital map, the locations within a certain range are considered as a single location called grid. The grid map is the map consist of $M \times N$ grids, where M and N depends on the research area and use case.

3.1.2 Grid-based Trajectory

Traditionally, the trajectory T is considered as a sequence of consecutive GPS points, i.e., $T = \{loc_{t_1}, loc_{t_2} \dots loc_{t_N}\}$, where loc_i is a pair containing the longitude,

latitude and timestamp information. According to the Definition 1, in this study, I use grid IDs to replace the GPS points in the former trajectory definition. Thus, the Grid-based Trajectory can be represented in this form:

$$T = \{Grid_{t_1}, Grid_{t_2} \cdots Grid_{t_i}, \cdots Grid_{t_N}\},\ i \in \mathbb{N},\ 1 \leq i \leq N \tag{3.1}$$

where the location point $Grid_{t_i}$ in the grid map contains the grid ID $Grid_i$ and the timestamp t_i. Furthermore, for each grid-based trajectory, the external factors such as the corresponding user/driver (*Driver_ID*), the day of the week (*Day_of_Week*), current speed (v) and traveled distance till now (*Distance*) are also recorded in the trajectory data. An intuitive example is shown in Fig.1.1.

3.2 Recommender System

3.2.1 Top-N Recommendation

In general, in the field of Machine Learning, recommendation is to use algorithms to predict the “ rating ” or “ preference ” a user would give to an item. These predictions are made by analyzing data about the user ’ s past behavior, as well as data about the items themselves. The goal of a recommendation system is to present users with items that they are likely to be interested in, based on their past behavior and preferences.

Let U be the set of all users and I be the set of all items. A recommendation function R can be defined as :

$$R : U \times I \rightarrow S \tag{3.2}$$

where S is the set of all possible scores that an item can have. This function takes

as input a user and an item and returns a score indicating how relevant that item is for that user. Based on the use case, recommendation can be divided into several categories[39], such as Collaborative Filtering (CF) based Recommendation, Content based Recommendation, Deep Neural Network (DNN) based Recommendation, etc. Each of these types has its own strengths and weaknesses, and the choice of which to use typically depends on the specific use case and the available data. In this paper, we focus on the **Top-N based** approach.

User-Item Matrix: Assume that M is a user-item matrix, where M is a matrix of m rows (users u) and n columns (items i). Each cell $M_{u,i}$ represents the rating/score that user u has given to item i. If user u has not rated item i, then $M_{u,i}$ is defined as 0.

Rating/Score Prediction Function: The recommendation function $R : U \times I \rightarrow S$ takes as input a user and an item and returns a score indicating the predicted preference of that user for that item. The nature of this function can depend on the specific algorithm used by the recommendation system (collaborative filtering, matrix factorization, deep learning based etc.).

Top-N Recommendation List: For a given user u, the top-N recommendation list, $L(u, N)$, is defined as the top N items from the set I with the highest scores $R(u, i)$ after sorting in descending order. Formally, we have:

$$\begin{gathered} L(u, N) = \{i_1, i_2, \cdots i_N\} \\ \forall j, \text{st.} i_j \in I, 1 \leq j < N, R(u, i_j) \geq R(u, i_{j+1}) \end{gathered} \tag{3.3}$$

The goal of a top-N item recommendation algorithm is to generate for each user a list of N items that the user will most likely be interested in, based on their

past interactions and potentially other factors such as user/item features, time, and context.

3.2.2 Deep Recommender System

Deep Recommender Systems represent the integration of deep learning methodologies into recommendation systems. The specific architecture of deep learning, the data incorporated, and the problem to be solved (such as rating prediction or top-N item recommendation) can significantly influence the configuration and process flow. Consequently, it is arduous to encapsulate all these varying methodologies within a unified framework. Therefore, in this section, I will only give a brief introduction of the Deep Recommender System utilizing Embeddings and Multi-Layer Perceptron(MLP). The Architecture of Embedding & MLP based Deep Recommender System can be formally defined as:

Denote $\mathcal{U}$ as the set of all users and $\mathcal{I}$ as the set of all items. Each user u in $\mathcal{U}$ and item i in $\mathcal{I}$ are represented as one-hot encoded vectors in a high dimensional space.

Embedding Layer: The one-hot encoded vectors are passed through an embedding layer to get dense vector representations for users and items. This layer can be seen as a lookup table that transforms sparse vectors into dense ones. Let E_u and E_i denote the embedding functions for user and item respectively. The output of these functions are $e_u = E_u(u)$ and $e_i = E_i(i)$, which are the embedding vectors of user u and item i.

Concatenation: The user and item embeddings are concatenated together to form

a single vector, i.e., $x = [e_u; e_i]$.

Multi-layer Perceptron: This concatenated vector is then passed through an MLP to capture complex interactions between user and item. The MLP consists of several fully connected layers and activation functions like *ReLU, Sigmoid, and so on*, and the output is another vector. Let MLP denote the MLP function, then $h = MLP(x)$ is the output vector.

Output Layer: The last fully connected layer is the output layer, which produces the final prediction. It's typically a single neuron with a linear or sigmoid activation function. Let O denote the output function, then the predicted rating $r_{ui} = O(h)$.

Training: The system is trained by minimizing a loss function, which measures the difference between the predicted and true ratings. Common choices for the loss function include Mean Squared Error (MSE) and Cross-Entropy.

In summary, a Deep Recommender System using embedding and MLP transforms sparse user and item vectors into dense embeddings, concatenates these embeddings, applies an MLP to learn high-level interaction features, and finally predicts ratings or preferences.

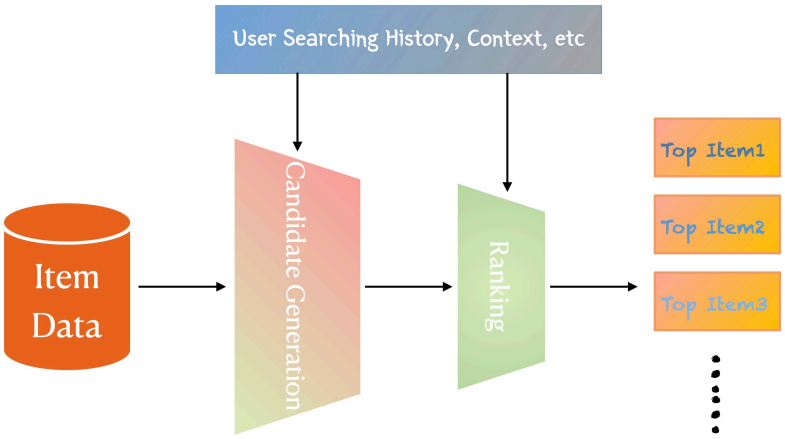


Fig. 3.1: This is a typical framework demo of the modern recommendation system. Here substitute the item with the grid in the grid trajectory prediction task and replace the history data with trajectory, then the framework of the method using recommender system based approach to predict the trajectory can be obtained. In this thesis, there is no intent to fulfill the whole architecture since this is just a pilot research to check the feasibility of proposed model.

3.3 Neural Network

The Neural Network is a broad concept in the field of Artificial Intelligence. It is the cornerstone technology of Deep Learning, a branch of Artificial Intelligence. Due to space constraints, I will only briefly introduce the neural network techniques relevant to this paper, such as Long Short-Term Memory networks and the Graph Embedding techniques.

3.3.1 Long Short-Term Memory

Long Short-Term Memory networks, LSTMs are designed to avoid the long-term dependency problem which traditional RNNs suffer from. They achieve this by incorporating memory cells and three types of gates: input, forget, and output gate. Let's denote the input at timestep t as x_t, the output as h_t, and the cell state

as c_t. LSTM updates the cell state and output by the following set of equations:

1. Forget gate: This gate decides what information should be discarded or kept. It is a sigmoid layer that takes h_{t-1} and x_t and outputs a number between 0 and 1 for each number in the cell state c_{t-1}. A 1 represents ”completely keep this” while a 0 represents ”completely get rid of this”.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.4}$$

2. Input gate: This gate updates the cell state with the new information. It has two parts. A sigmoid layer called the ”input gate layer” decides which values we’ll update, and a tanh layer creates new candidate values that could be added to the state.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{c}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{3.5}$$

3. Cell state: It is a combination of the old state (scaled by forget gate value) and the new candidate values (scaled by input gate value).

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{3.6}$$

4. Output gate: This gate decides what the next hidden state should be. The hidden state contains information about previous input data, and is used for predictions. It is a filtered version of the cell state, and the filter is decided by the input data.

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(c_t) \end{aligned} \tag{3.7}$$

Here, $*$ denotes element-wise multiplication, σ is the sigmoid function, and the

Ws and bs are parameters to learn.

3.3.2 DeepWalk

DeepWalk, an algorithm developed by Perozzi et al.[40], is an effective method to learn continuous vector representations of vertices in graphs. The central concept behind DeepWalk is performing random walks over a graph to generate sequences of vertices, similar to sentences in natural language. These sequences are then projected onto a latent space using techniques derived from the natural language processing, specifically the Skip-gram model from Word2Vec.

Let us denote a graph as $G(V, E)$, where V represents the set of vertices and E symbolizes the set of edges. The primary goal of DeepWalk is to identify a function $\phi : V \rightarrow \mathbb{R}^d$, which maps vertices to a d-dimensional latent space. This process comprises two stages:

1. **Random Walks**: For every vertex $v \in V$, DeepWalk generates γ independent random walks of a fixed length l. Each random walk $W_v = \{v_1, v_2, \ldots, v_l\}$ is a sequence of vertices where the ith vertex is randomly chosen from the neighbors of the $(i-1)$th vertex.

2. **Embedding Learning**: The Skip-gram model, a context prediction model from natural language processing, is used to generate embeddings from the 'sentences' of vertices created by random walks.

The objective of the Skip-gram model is to identify vector representations that are suitable for predicting the surrounding vertices in a sequence for a given vertex.

This can be formally defined as the following optimization problem:

$$\min_{\phi} -\frac{1}{|V|} \sum_{v \in V} \log \Pr\left(W_v \mid \phi(v)\right) \tag{3.8}$$

where $\phi(v)$ represents the d-dimensional representation of vertex v and $\Pr(W_v|\phi(v))$ is the probability of walk W_v given the vector representation of vertex v.

The probability $\Pr(W_v|\phi(v))$ is factorized using the conditional independence assumption and then approximated using the softmax function:

$$\Pr\left(v_i \mid \phi(v)\right) = \frac{\exp\left(\phi\left(v_i\right) \cdot \phi(v)\right)}{\sum_{v_j \in V} \exp\left(\phi\left(v_j\right) \cdot \phi(v)\right)} \tag{3.9}$$

for each vertex v_i in the random walk W_v. This objective function is typically optimized using Stochastic Gradient Descent (SGD) or similar optimization methods. The result of the DeepWalk algorithm is a series of d-dimensional embeddings for each vertex in the input graph. These embeddings effectively capture the local neighborhood structure surrounding each vertex. It should be noted that DeepWalk focuses on first-order proximity between vertices and does not consider higher-order or global structural properties of the graph.

3.4 Problem Definition

Grid Trajectory Prediction. In this study, the problem being addressed can be formally defined as: under a certain sampling frequency f, given a trajectory $T = \{Grid_{t_1}, Grid_{t_2} \cdots Grid_{t_i}, \cdots Grid_{t_{N-1}}\}$, predict $Grid_{t_N}$ by providing K possible candidates which have a descending probability.

Chapter 4

Methodology

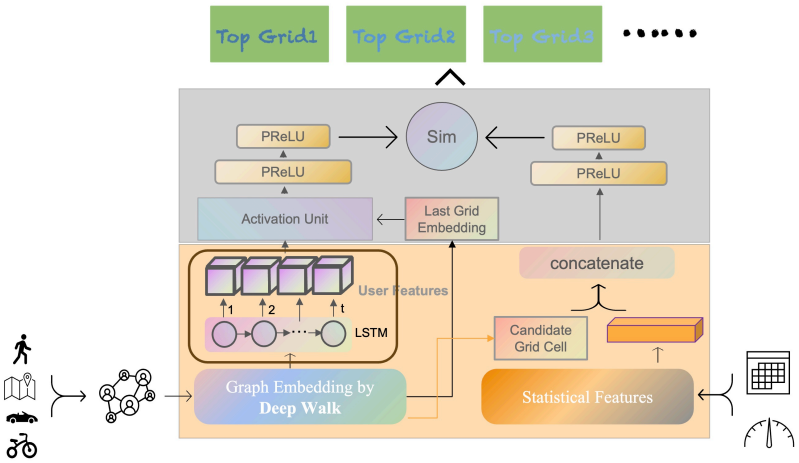


Fig. 4.1: RSTP Architecture

In this section, I will present the architecture of the proposed model, Recommender System based Trajectory Prediction model, RSTR. And provide a detailed explanation of each functional components in RSTR. As shown in Fig.4.1, RSTP is consist of two modules as a typical Recommender System do: the Embedding Construction module and the Candidate Generation module. In each module, it contains several functional components to deal with different information. More

specifically, Embedding Construction module is comprised of three components: Temporal Component, Spatial Component and Attribute Component. And the Candidate Generation module is comprised of two components: Interaction Component and Output Layers. As the detailed explanation of each component will be presented in each subsection.

4.1 Embedding Construction

The Embedding Construction module is designed to transform the original sparse trajectory data into dense representations, represented as high-dimensional vectors/tensors. Once the representation for each grid cell and individual trajectory is obtained, similarity computation can be easily conducted. As a result, processes such as collaborative filtering or recommendation generation can be carried out seamlessly based on the previously computed similarities. The primary challenge in extracting these dense representations lies in the simultaneous consideration of complex spatiotemporal information and other external factors in conjunction with the original trajectory data. To address this challenge, the Embedding Construction module is comprised of three distinct components: the Temporal Component, the Spatial Component, and the Attribute Component. Each of these components plays a crucial role in processing and integrating different types of information, thereby enhancing the effectiveness and accuracy of the subsequent trajectory prediction.

4.1.1 Attribute Component: Embedding and Concatenation

As previously mentioned, trajectory prediction is influenced by a myriad of complex factors, such as transportation mode, day of the week, and driving habits. To encapsulate such factors into the proposed model, I designed a component I

refer to as the Attribute Component. As illustrated in Fig.4.1, I have integrated attributes such as transportation mode (bus/walk/bike etc.), driver ID, time information (day of the week and time slot of the sampling point), and other statistical features (current speed/average speed/distance traveled, etc.). These attributes are respectively denoted as modeID, driverID, weekID, and timeID. It is important to note that these factors are categorical values and cannot be directly fed into the neural network. To address this, in our proposed model, I adopt the embedding method proposed by Gal and Ghahramani [41]. This method transforms each categorical attribute into a low-dimensional real vector, which can then be processed by our neural network model effectively. Specifically, the embedding method maps each categorical value $c \in C$ to a real space $R^{E\times 1}$ (also known as the Embedding space) by multiplying it with a parameter matrix $W \in R^{C\times E}$. Here, C signifies the vocabulary size of the original categorical value and E represents the dimension of the embedding space. Usually, E is significantly less than C ($E \ll C$). When compared with one-hot encoding, the embedding method has two main advantages. Firstly, as the vocabulary size of the categorical values can be quite extensive (for instance, there are over 2000 drivers in our dataset), the embedding method effectively reduces the input dimension, thus offering computational efficiency. Secondly, it has been demonstrated that categorical values with similar semantic meanings are usually embedded into close locations [41]. This implies that the embedding method facilitates the identification and sharing of similar patterns across different trajectories. This inherent property can be highly advantageous in improving the predictive accuracy of our model.

Besides the embedded attributes, we further incorporate other important statistical attributes, like the distance traveled, current speed and average speed. Let's use $Dis_{i,t}, v_{i,t}, \bar{v}_{i,t}$ to denote the total traveled distance from the starting point to current

locations, current speed and the average speed from start till now. Formally, those attributes can be computed as:

$$\begin{aligned} Dis_{i,t} &= \sum_{j=0}^{t-1} dist(Loc_{i,j}, P_{i,j+1}) \\ v_{i,t} &= \frac{dist(Loc_{i,t}, Loc_{i,t-1})}{\Delta t} \\ \bar{v}_{i,t} &= \frac{Dis_{i,t}}{t} \end{aligned} \tag{4.1}$$

where *dist* is the geographic distance between two GPS points. Then, we concatenate the obtained embedded vectors together with these statistic features. The concatenation is used as the output of the attribute component, denoted by *attr*.

4.1.2 Spatial Component: DeepWalk

As previously noted, DeepWalk is a technique that extracts high-dimensional vector representations of nodes from a graph. This is accomplished through the emulation of random walks over the graph, which effectively produces 'sentences' of nodes. These sequences are then embedded into a latent space by employing techniques derivative of natural language processing. In particular, the Skip-gram model from Word2Vec[42]. Through this mechanism, DeepWalk generates nuanced, contextually-informed embeddings for each node within the graph, effectively capturing the localized neighborhood structure for individual nodes.

To utilize the DeepWalk method, it is necessary to construct a graph from the map data. Since the proposed model uses a Grid Map, it is intuitive to perceive the grid map as a spatial graph. More specifically, we consider each grid cell g_i as a vertex v in the graph, with the connections among grid cells representing the edges e in the graph. The connection between two grid cells is established if

there are any trajectories traversing from grid g_x to another grid g_y. And the more links established between two grids, the higher weight w the two grids have. This construction of the graph effectively maps spatial relationships between locations onto a topological structure, facilitating the subsequent application of DeepWalk for the extraction of grid cell representations.

Before apply the DeepWalk method, according to the trajectory data and map information, construct a weighted undirected graph $\mathcal{G}$, where the vertex is the grid cell g_i, and the edge e_i is the times a certain trajectory across any two grid cell sequentially. Then, the DeepWalk method can be applied to extract the high-dimensional grid cell representations $grid_i$.

4.1.3 Temporal Component: LSTM

Indeed, the first reason why LSTM network is that the application of LSTM networks for trajectory prediction tasks is grounded in their demonstrable ability to handle and model non-linear temporal dependencies inherent in sequential data. The underlying premise of these applications lies in treating trajectories as a series of interconnected data points ordered in time. Another reason why LSTM network is necessary in this work is that the raw trajectory data is greatly unequally distributed (detailed explanation will be introduced in the Chapter 5). To mitigate the impact of this variable on the prediction results, LSTM serves as an apt choice, given its capability to transform sequential data of arbitrary lengths into embeddings of a uniform length. This characteristic is particularly salient for our task, as it accommodates the variable-length trajectories in our data. Given the potent capabilities of LSTM models in dealing with time-series data, it is entirely rational to leverage them for trajectory data. However, the use of LSTM in this work deviates from traditional approaches. Instead of directly employing LSTM

for predicting subsequent locations for a given trajectory, a task that can potentially be hampered by an overwhelming number of possible outcomes, LSTM is utilized for extracting high-level embedded trajectory representations.

Based on the extracted grid cell representations $grid_i$, it is easy to obtain the vector representation of a trajectory, since the sequential combination of high-dimensional grid cell vector $grid_i$ is a kind of embedded trajectory representations $\mathcal{T}_i' = \{grid_{i_0}, grid_{i_1}, \cdots, grid_{i_t}\}$. However, as the raw data does not time-equally sampled (detailed reasons will be provide in Chapter 5), the performance of proposed model will be greatly affected by the number of sampling points, even failure to work. Thus, the output trajectory representation should be calculated as:

$$\mathcal{T}_i = \text{LSTM}(grid_{i_0}, grid_{i_1}, \cdots, grid_{i_t}) \tag{4.2}$$

Again, $\mathcal{T}_i$ is not the prediction result of the next location. It is the embedded trajectory vector, more specifically, is the combination of hidden state (3.5) and cell state (3.5). The hidden state and cell state are intermediate vectors generate by the LSTM network, the vectors means characterization of the previous time-step ' s data (short-term memory) and the information from all previous time-steps that have been processed (long memory) respectively.

4.2 Candidate Generation

Through the Embedding Construction module, the embedded vectors of both trajectories and grid cells are obtained. These vectors allow for a direct computation of the similarity between trajectories and grid cells. Up to this point, the proposed model has taken into account temporal factors, spatial factors, and attribute factors. However, in real-world scenarios, temporal and spatial factors typically

influence each other, indicating that modeling their correlations and dependencies is also vital for accurate prediction results. Consequently, before inputting the high-level feature representations into the Multi-Layer Perceptron or Deep Neural Networks (DNN)[38], an Activation Unit is incorporated to facilitate interactions between temporal features and spatial features.

4.2.1 Interaction Component: Activation Unit

This component, called an Activation Unit, is used to model feature interactions, which play an essential role in generating predictions. It utilizes a mini-network structure with a hidden layer, enabling the learning of sophisticated feature interactions automatically rather than relying on manual feature engineering. The model learns weights that emphasize meaningful interactions while diminishing the impact of less relevant ones, refining the prediction accuracy. In this these, the Activation Unit learn the weights between trajectory and candidate grid cells automatically. This way, the Activation Unit promotes better prediction results through the incorporation of interrelated temporal and spatial features.
In detailed, activation units are applied on the grid trajectory features, which performs as a weighted sum pooling to adaptively calculate trajectory representation $\mathcal{T}_U$ given a candidate grid cell g_c as:

$$\mathcal{T}_U(g_c) = f\left(g_c, grid_1, grid_2, \ldots, grid_H\right) = \sum_{j=1}^{H} a\left(grid_j, g_c\right) grid_j = \sum_{j=1}^{H} w_j \cdot grid_j \tag{4.3}$$

where $\{grid_1, grid_2, \ldots, grid_H\}$ is the list of embedding vectors of the past traveled girds of user U with length of H. $\boldsymbol{g_c}$ is the embedding vector of candidate $\boldsymbol{c}$, and $\boldsymbol{a}(\cdot)$ is a feed-forward network with output as the activation weight as shown in Fig.4.2. In this way, $\mathcal{T}_U(g_c)$ varies over different candidate grid cells. In a more

intuitive sense, the current trajectory embeddings reflect dynamic variations, subject not only to changes in temporal information (trajectory) but also in response to alterations in spatial features. This multi-dimensional variability lends a robustness and complexity to the model that mirrors real-world conditions more accurately. The correlations and dependencies between spatial and temporal factors are intertwined, thereby generating trajectory embeddings that offer a more comprehensive understanding of the movement patterns.

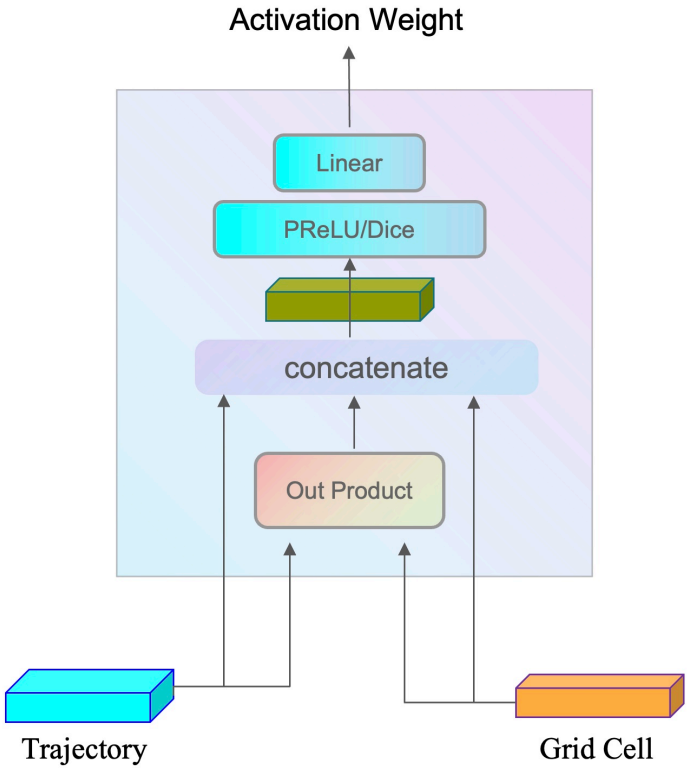


Fig. 4.2: The architecture of activation unit.

4.2.2 Multi-Layer Perceptron

The MLP is most common layer to conduct the non-linear transformation of input data, which make the prediction result more robust. And the MLP is also known

as a typical type of Feed-Forward Network (FNN). Let's denote the weights from layer i to layer $i + 1$ as $\mathbf{W}_i$ and the bias at layer $i + 1$ as $\mathbf{b}_{i+1}$. If $\mathbf{h}_i$ represents the output of layer i and f is the activation function, then the output of layer $i + 1$ can be represented as:

$$h_{i+1} = f(W_i \mathbf{h}_i + b_{i+1}) \tag{4.4}$$

This formula represents the feed-forward computation for a single layer. In an MLP with L layers, this computation would be repeated L times (once for each layer). The output for the final layer L is typically passed through a final activation function to produce the final output of the network. In this thesis, the activation function called **Dice** is used. Zhou et al.[38] developed the novel activation function based a classical activation funtion **PReLU**, which aims to deal with different distribution problem similar to the one I countered in this studies. Mathematically, the PReLU function can be formulated as:

$$f(s) = \begin{cases} s & \text{if } s > 0 \\ \alpha s & \text{if } s \leq 0 \end{cases} = p(s) \cdot s + (1 - p(s)) \cdot \alpha s \tag{4.5}$$

Wherein s represents a dimension of the input to the activation function $f(\cdot)$, and $p(s) = I(s > 0)$ is an indicator function that dictates the behavior of $f(s)$, permitting it to switch between the channels $f(s) = s$ and $f(s) = \alpha s$. In the latter channel, α serves as a trainable parameter. We refer to $p(s)$ as the control function. The control function of the PReLU activation function is depicted on the left side of Fig. 4.3. PReLU sets a rectified point at a value of 0, an approach that may be ill-suited when the inputs across different layers display diverse distributions. Given this consideration, we propose a novel data adaptive activation

function dubbed Dice to address this issue in a more sophisticated manner.

$$f(s) = p(s) \cdot s + (1 - p(s)) \cdot \alpha s, p(s) = \frac{1}{1 + e^{-\frac{s - E[s]}{\sqrt{Var[s] + \epsilon}}}} \tag{4.6}$$

The control function for the Dice activation function is illustrated on the right side of Fig. 4.3. From one perspective, Dice can be considered as a generalized form of PReLU. The central idea underpinning Dice is to adaptively modify the rectified point in accordance with the distribution of the input data, with the value being set to the mean of the input. Furthermore, Dice facilitates a smooth transition between the two channels. In the case where $E(s) = 0$ and $Var[s] = 0$, Dice reduces to the PReLU function.

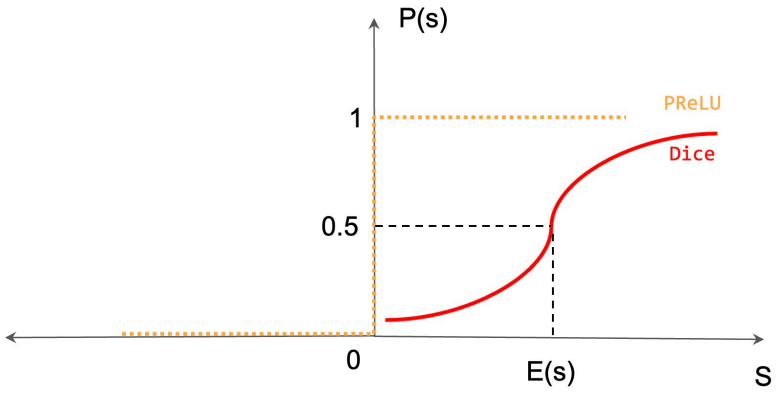


Fig. 4.3: Control function of PReLU and Dice.

Finally, get through MLP layers, the cosine similarity is used to measure final result, which is the possibility of the candidate grid cell to be the next location for a given trajectory.

Chapter 5

Experiments

In this section, I will give all the details of experiments settings and How I conducted the experiments. After that, the performances comparison of different models on the same dataset is provided as well.

5.1 Experiments Settings

5.1.1 Dataset Description

I evaluate the proposed model on two real-world datasets:

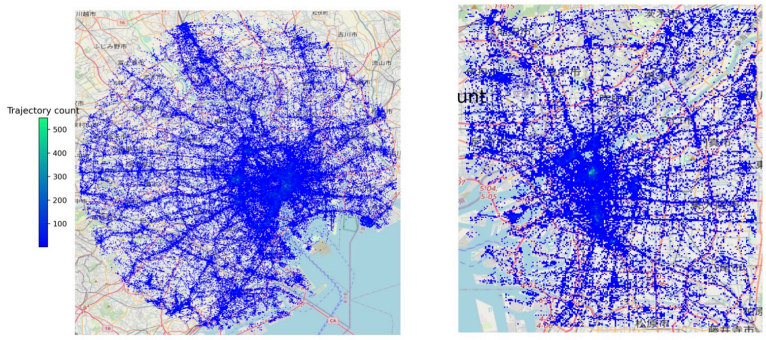
- **Tokyo Dataset:** Tokyo dataset comprises 612,010 trajectories, encompassing over 120 million GPS records. These data were collected from 2,319 users over a period from August 23rd to August 30th, 2022, within the urban confines of Tokyo, Japan. The shortest trajectory only contains 13 GPS records (2km) and the longest trajectory contains 2109 GPS records (41km). Besides, the Tokyo dataset contains 5 different transportation modes, including walk, bike, car, bus and train. Apart from the bike trajectories, which comprise approximately 12% of the total dataset, a proportion slightly

less than other modes, the remaining transport modes are approximately equally distributed throughout the dataset. This means that, despite the slight underrepresentation of bike trajectories, a broad diversity of transport modes is still well represented in the dataset, providing a robust basis for the generalization of our trajectory prediction model across different transportation contexts.

- **Osaka Dataset:** Osaka dataset is comprised of 368,216 trajectories (over 70 million individual GPS records), which is collected from 10,53 users over a period from August 23rd to August 30th, 2022, within the urban confines of Osaka, Japan. The shortest trajectory only contains 4 GPS records (0.4km) and the longest trajectory contains 1349 GPS records (28km). Same like the Tokyo dataset, Osaka dataset also contains 5 different transportation modes, including walk, bike, car, bus and train. Additionally, each mode of transportation is approximately equally distributed throughout the dataset.

As discernible from the dataset description, it is conspicuous that the distribution of GPS data points is markedly uneven, ranging from 10 to 2000, and the intervals between each sampling point are not necessarily consistent. In order to mitigate potential detrimental effects on prediction accuracy engendered by these factors, the original data underwent preprocessing. This involved a resampling process, which was conducted based on a defined time interval, thereby ensuring uniformity and reducing potential bias in our analyses.

Fit 5.1 gives an intuitive example how those dataset looks like. Is important to mention that the research area of both two dataset are square, like from 139.55 to 139.85 and 35.55 to 35.85. The reason why the Tokyo dataset visualized in a circle is that sampled data is a little bit larger than the research area to make sure the data used in the experiment will not be corrupted by accident, for example



Tokyo Dataset **Osaka Dataset**

Fig. 5.1: Visualization of two datasets on the grid map.

only a part of GPS points of a entire trajectory is used.

Map Division. In addition, due to the divergent distributions of the two datasets, the research areas considered differ slightly. Specifically, for the primary experiments conducted on the Tokyo dataset, the selected area is demarcated by longitude and latitude values ranging from 139.55 to 139.85 and 35.55 to 35.85, respectively. And in Osaka, the selected area is demarcated by longitude and latitude values ranging from 135.35 to 135.65 and 34.55 to 34.85. To make the consistence with the default model, the grid size is set to be $100 \times 100\,\text{meter}^2$ both in Tokyo and Osaka. It is important to mention that other map information like Point-Of-Interest (POI) data are not considered in the prototype model, since the main purpose is to verify the functionality of recommender system based approach on the trajectory prediction task.

5.1.2 Parameter Settings

The parameters I used in the following experiments are described as follows:

- In the attribute component, the embedding vector dimension is chosen like

weekID to R^4, timeID to R^{16}, driverID to R^{12}.

- In the spatial component, the LSTM is implemented by TensorFlow, and the embedding dimension (units) is set to be R^{64}, the used *activation* function is *tanh*, and recurrent activation is *sigmoid*. In order to consistent with the output of DeepWalk, the single layer LSTM network is more suitable.

- In the temporal component, the parameters of a DeepWalk model implemented by Python is like number of random walks per node (*num_walks*) is 10, the length of random walk (*walk_length*) is 100, the dimension of embeddings (*representation_size*) is $R^6$4, the window size (*window_size*) is 5, the number of epochs (*num_iter*) is 10, and the learning rate (r) is 0.002.

- As for the physical devices and software versions, the propose model is implemented with TensorFlow 2.11.0, a widely used Deep Learning Python library. The versions of other used software/library is Pandas 1.5.2, Numpy 1.24.1 and faiss 1.7.2, where faiss is an open-source library developed by Facebook AI that provides efficient and scalable methods for handling similarity search and clustering of high-dimensional vectors. The hardware involed is 2 GeForce RTX 2080 Ti GPU, 1 Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz CPU, and the developing environment is Ubuntu 20.04.6 LTS.

5.1.3 Metrics

In the following experiments, I used two metrics to measure the performance of the proposed model, i.e., Cosine Similarity and Hit Rate. During the model training process, Cosine Similarity is used to measure the quality of generated embeddings. And in the predicting process, Hit Rate is used to evaluate the overall

performance of the proposed model.

Cosine Similarity. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It is defined as the cosine of the angle between the two vectors, which is determined by the dot product of the vectors divided by the product of the vectors' magnitudes. In our case, the Cosine Similarity is used to compute the similarity of embedded trajectory vector and the candidate embedded grid cell vectors. Mathematically, cosine similarity is computed as follows:

$$cos(\Theta) = \frac{\mathcal{T}_i \cdot g_j}{\|\mathcal{T}_i\| \cdot \left\|g_j\right\|} \tag{5.1}$$

where $\mathcal{T}_i \cdot g_j$ denotes the dot product of embedded trajectory vector $\mathcal{T}_i$ and the embedded grid vectorg_j, and $\mathcal{T}_i$ and g_j represents the magnitudes (or lengths) of vectors $\mathcal{T}_i$ and g_j. The result of this equation is a value between -1 and 1. Basically, the larger cosine similarity, the more similar two vectors. In this thesis, a larger cosine similarity implies a greater similarity between the two vectors. In the context of our trajectory prediction task, this translates to the higher likelihood of a grid being the next point in a given trajectory. When considering the recommendation of the next grid cell based on this principle, the grid cells with larger cosine similarity values to the trajectory's current direction are more likely to be the subsequent point in the trajectory.

Hit Rate. The metric used to measure the overall performance is called Hit Rate. Hit Rate is a commonly employed evaluation metric used to assess the system's effectiveness. It quantifies the proportion of true positive recommendations out of all the recommendations made by the system. Formally, Hit Rate is defined as the number of hits divided by the total number of recommendations given. In our case, the ground truth or hit is the grid which a certain user will appeared in next

timestamp, it can formulated as:

$$\text{HitRate} = \frac{number\ of\ hit}{number\ of\ trajectories}$$

where *hit* means that the ground truth grid cell appeared in the top-K recommendations. It important to point out that unlike the recommendation task, there is only one location can be ground truth for a given trajectory at certain time.

5.2 Performance Comparison

To demonstrate the strength of the proposed model. I first compare the proposed model with several baseline methods and analysis the difference among the experiment results.

5.2.1 Competitors

To best of my knowledge, there is no similar approach, for instance, using the recommender system approaches to predict individual trajectory. Thus, the baseline mode is select from the classical models that can applied to the grid trajectory prediction task, also including the simplified proposed model. The detail follows:

AN: AN stands for Average Neighbour, the idea is to just select the neighbour grid cells of the last appeared location for a given trajectory as the predict results. For example, Since the map is already divided by the grid, it is clear that the neighbour grids are the grids which are most close to the target grid.

LSTM: As outlined in Chapter 3, the LSTM model is a traditional and effective approach for sequential data prediction. Given that trajectory data, even when rep-

resented as grid trajectories, is essentially sequential data, it stands to reason that LSTM would be used to predict the next location for a given trajectory. Nevertheless, it is important to note that the standard LSTM model was not originally designed for recommendation purposes. Consequently, in subsequent experiments, we only consider the top-1 recommendation for LSTM. This is easily understood when considering that the prediction itself can also be viewed as the top-1 recommendation.

YouTubeDNN: YouTubeDNN refers to the model proposed in the paper named "Deep Neural Networks for YouTube Recommendations[37]". Actually, the proposed model is derived from the YouTubeDNN model. And the YouTubeDNN is a famous model in the field of recommendation, which is designed to solve the efficient recommendation problem for the famous video website YouTube. In the grid trajectory case, it is also can be used for the trajectory prediction.

Temporal RSTP: Temporal RSTP is a simplified model of the proposed RSTP model, or to put it simply, it can be regard as a simple combination of YouTubeDNN and LSTM. In the temporal RSTP model, only the temporal information is considered. That is to say, the LSTM model is used to extract the temporal information behind the ordinary trajectory. As for the spatial embedding of the grid cell, it is just using a function provided by the TensorFlow to random generate a dense embedding representation of any numerical value like Grid ID.

Spatial RSTP: Spatial RSTP is also simplified model of the proposed RSTP model, similarly, it can be regard as a simple combination of YouTubeDNN and DeepWalk. Similar to the temporal model, the spatial temporal model only consider the spatial information by utilize the DeepWalk model to capture the spatial

information. To avoid the failures caused by the variant-length of trajectory, after obtaining the embedded grid cells, we directly compute the average $R^6 4$ vector of past traveled grid cell as the representation of trajectory.

Datasets	Methods	K@1	K@10	K@100
Tokyo	AN	22.98% (9)	25.46% (25)	**40.71%(81)**
	LSTM	0.11%	-	-
	YouTubeDNN	0.76%	3.92%	9.74%
	Temporal RSTP	23.37%	25.58%	30.68%
	Spatial RSTP	1.23%	8.32%	19.39%
	RSTP (proposed)	**23.66%**	**28.33%**	34.76%
Osaka	AN	18.59%(9)	26.66%(25)	48.37%(81)
	LSTM	0.13%	-	-
	YouTubeDNN	0.18%	3.32%	6.84%
	Temporal RSTP	20.18%	28.32%	29.23%
	Spatial RSTP	1.49%	6.32%	20.33%
	RSTP (proposed)	**25.77%**	**30.43%**	**48.53%**

Table. 5.1: Performance Comparison (15 minutes interval)

Table 5.1 shows the experiment results. As we can see, simply using the Average Neighbour method has a relative good results comparing to the ordinary unoptimized method like LSTM, YouTubeDNN, even superior to the Spatial RSTP method when the recommendation number K is small. One possible reason why LSTM model almost fail to work is that as the multi-classification problem, there is too many possible results to choose for the LSTM model. Additionally, LSTM model can not get enough information to distinguish those classes (grids). As for the YouTubeDNN model, comparing to the LSTM, its performance is similar to LSTM. With the k increasing, the result is also increasing rapidly. We can say that this result indicates that the recommender system approaches work for the grid trajectory prediction task.

In general, the proposed model have a good performance in most the cases. As the

incorporation of spatiotemporal component, the performance is greatly improved, which indicated the importance of the spatiotemporal information in the trajectory prediction task. However, as the K or the number of neighbours increase to a certain level, the performance of naive Average Neighbour method may Superior to the proposed method. That is because in the trajectory predictions, the physical constrains exists. When the number of neighbours increases to a certain level, it will easily cover all possible results like a brute force approaches. While the proposed methods works in a completely different logic, and it is more promising comparing to the Average Neighbour method. The detailed explanation will be given in the **Case Study** section.

5.3 Case Study

In this section, I will introduce some detailed case study to show the strength of the proposed model and explain why this approach is more promising than the brute force method like Average Neighbour method.

5.3.1 Case Study 1: Effect of Attribute Component

We validate the effectiveness of various attributes, including timeID, driverID, and weekID, through a series of controlled experiments conducted on the Tokyo Dataset. For each experiment, we selectively eliminate one attribute. Our findings reveal that both timeID and weekID have a considerable impact on the Hit Rate. The elimination of these two attributes leads to a decrease in the Hit Rate of **0.79%** and **1.07%** in the Top-1 scenario, respectively. This result aligns with our intuitive understanding, that is, individuals typically exhibit distinct trajectory patterns at different times of the day and on different days of the week. For instance, during working hours, most people are likely to remain in their homes or offices,

resulting in stable trajectories. Conversely, on weekends, people may travel to different places compared to weekdays. The removal of the DriverID leads to a minor decrease of **0.15%** in the Top-1 scenario, which might not seem substantial. However, if both the DriverID and statistical features such as speed and distance are eliminated, the decrease increases to **0.9%** in the Top-1 scenario. These experiments underscore the importance of attribute information for prediction results, particularly when K is small.

5.3.2 Case Study 2: Effect of Spatiotemporal Component

Table 5.1 vividly illustrates that the independent incorporation of both Spatial Component and Temporal Component into the YouTubeDNN model significantly enhances its performance, albeit with differing growth patterns. For clarity, we have collated the results into Table 5.2. Here, the method termed 'combined' refers to the numerical amalgamation of the results from Temporal RSTP and Spatial RSTP. More specifically, a union operation is performed on the prediction result sets of Temporal RSTP and Spatial RSTP, following which the hit rate is recalculated. The results are then documented in Table 5.2. Three notable findings can be extrapolated from Table 5.2:

- The proposed temporal component serves to enhance the overall performance of the prediction model. Interestingly, the degree of this improvement does not show substantial sensitivity to the increase in the number K. One plausible explanation for this observation is that the temporal component predominantly captures time-related information, such as the trajectory of the previous day. Given that most individuals exhibit relatively consistent trajectory patterns within a week, an increase in K may not necessarily alter the results provided by the model. This is reflected in the incremental improvement not showing substantial sensitivity to the rise in the value of

K.

- The proposed spatial component also enhances the overall performance of the prediction model, showing considerable sensitivity to increases in the number K. Given that the spatial component is designed to identify similar grid cells, it is expected to find numerous similarities at the city-scale. For instance, areas surrounding different stations may exhibit striking similarities. As such, while the performance in the Top-1 case may not meet expectations, an increase in K can yield a larger set of similar grids. These are likely to contain the ground truth, hence improving the model's performance as K increases.

- Another intriguing observation is that the proposed RSTP model consistently outperforms the numerical combination of the Spatial and Temporal models in all conducted experiments. Particularly in the Osaka dataset, with Top-100 recommendations, the performance of RSTP is approximately 14% superior to that of the numerical combination. This suggests that the interaction component in the 'candidate generation' phase of the model is functioning effectively, enabling it to capture the nuanced interplay between the spatial and temporal factors.

Datasets	Methods	K@1	K@10	K@100
Tokyo	Temporal RSTP	23.37%	25.58%	30.68%
	Spatial RSTP	1.23%	8.32%	19.39%
	RSTP	**23.66%**	**28.33%**	**34.76%**
	Combined	23.37%	25.90%	31.48%
Osaka	Temporal RSTP	20.18%	28.32%	29.23%
	Spatial RSTP	1.49%	6.32%	20.33%
	RSTP	**25.77%**	**30.43%**	**48.53%**
	Combined	20.33%	29.42%	34.53%

Table. 5.2: Verification of the functionality of Spatiotemporal Component.

5.3.3 Case Study 3: Potential Analysis of RSTP.

Before proceeding with the analysis of performance across different transportation modes, it is imperative to first examine the results of a comparable experiment conducted in Case 2. In this experiment, we compare the proposed model with the brute force method known as Average Neighbor. Look at table 5.3, a different finding is that the numerical combination got much better performance, which means those prediction provided by those two method are not completely the same. More specifically, as shown in table 5.3, around **34%** of the prediction results in each method is independent since incremental of method **Combined** is roughly one third comparing the method **AN** and method **RSTP**.

One plausible explanation for this result is as follows: The brute force method, Average Neighbor, considers all possible results without taking into account the physical constraints of the trajectory prediction task, such as the speed of transportation modes and road connections. In contrast, the trajectory prediction task is inherently constrained by these physical conditions, resulting in a limited range of potential outcomes within the vicinity of the last observed location. Therefore, in this particular scenario, the brute force method can still achieve relatively good performance due to the proximity constraint. However, the proposed RSTP method, which aims to identify grid cells similar to the last observed grid or the entire trajectory, focuses on finding more accurate predictions by leveraging the inherent patterns and similarities in the data. Considering these factors, combining the numerical results of the two methods can lead to an incremental improvement in accuracy compared to the individual base methods. By incorporating the strengths of both approaches, the combined method can potentially enhance the prediction performance by effectively capturing the relevant spatial and temporal patterns present in the data. If the physical constrains can be incorporated to the

model, a further improve mend on the prediction accuracy can be expected. I leave it as an intriguing direction for future work.

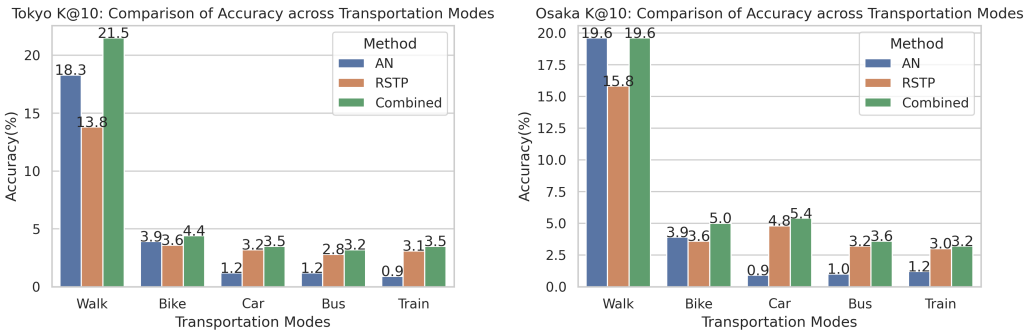
Datasets	Methods	K@1	K@10	K@100
Tokyo	AN	22.98% (9)	25.46% (25)	40.71%(81)
	RSTP	23.66%	28.33%	34.76%
	Combined	**30.71%**	**36.09%**	**48.61%**
Osaka	AN	18.59%(9)	26.66%(25)	48.37%(81)
	RSTP	25.77%	30.43%	48.53%
	Combined	**31.33%**	**36.83%**	**51.66%**

Table. 5.3: Functionality Analysis of Proposed Model.

Then let's take a deeper look at the performance of the two methods on different transportation modes. Fig.5.2 illustrates the performances among different transportation modes on Tokyo dataset and Osaka dataset. An observation is that, both in Tokyo and Osaka, the Average Neighbour method have a much better performance on the walk mode than the proposed method, and in the bike mode, those two method have similar performance. As for the left transportation modes, the proposed RSTP method is superior to the brute force AN method. This observation is consistent with the previous analysis that due to the physical constrains, the transportation mode with low speed leads to the the possible results constrained to the nearby area of last appeared location. That is, results of the walk mode and the bike mode provided by the AN method are better than the proposed model.

While the average speed increased, such as the car mode, bus mode and train mode, during 15 minutes, it can easily traveled 10 km away from the last locations. In this case, if using AN method to predict the next grid cell, the number of neighbour should reach to around $100 * 100$ grids to include the ground truth in all situations, which makes no sense in the real world applications. In contrast, the RSTP model give a relative better results than the brute force AN method. Even

just in the top-10 recommendation, it have the hit rate with a average 3% in such high speed situations. This may the contributions of the spatiotemporal component as we analyzed in **Case Study 2**.



(a) (b)

Fig. 5.2: Detailed performance comparison on transportation modes. (a) Tokyo Dataset (b) Osaka Dataset

In a summary, based on the extensive analysis on the 3 case studies, the functionality of the proposed model can be ensured through the conducted experiments. Moreover, the basic idea of the proposed model is different from the brute force method like *Average Neighbour*, and we can know that it is more practical in the real world applications through the analysis of *Case Study 3: Potential Analysis of RSTP*. However, the limitations of the proposed model are obvious as well, such as no considerations of physical constrains, lack of module to process the long-term temporal information. Till now, the feasibility of the recommender system approaches on the trajectory prediction task is approved, and the further improvement can be expected. I leave it as an intriguing direction for future work.

Chapter 6

Conclusion

In this thesis, a Recommender System based Trajectory Prediction (RSTP) model is proposed to address the grid-based trajectory prediction problem, striking a balance between prediction accuracy and the scope of the research area. The effectiveness of this modified Recommender System approach, which specializes in modeling spatiotemporal information, has been substantiated through extensive experiments, reinforcing the practicality of applying a Recommender System framework to the trajectory prediction problem. From a comprehensive literature review and the empirical investigations, it is observed that the proposed model exhibits clear practical utility and shows potential to forge a new approach for tackling trajectory prediction tasks. While the results may not be superior in all scenarios, the case studies suggest that this approach could potentially be a state-of-the-art technique, particularly when larger and more diverse datasets are taken into consideration. In future work, I will incorporate more context information such as the Point-Of-Interest (POI) information and long-term temporal information into the model to further improve the prediction performance.

Acknowledgements

First of all, I would like to thank the all the teachers and students in University of Tokyo for their assistance both in academic and in life. Secondly, I also have to express my gratitude to my dear friends in University of Tokyo, without of their help, I definitely can not gain such achievements and stand here finishing my master thesis. Thirdly, thanks the contributors of the extraordinary LaTeXtemplate. Finally, I would like to offer my sincere tribute to University of Tokyo, my alma mater and my parents!

Bibliography

[1] B. Cheng, X. Xu, Y. Zeng, J. Ren, and S. Jung, "Pedestrian trajectory prediction via the social-grid lstm model," *The Journal of Engineering*, vol. 2018, no. 16, pp. 1468–1474, 2018.

[2] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "Stgat: Modeling spatial-temporal interactions for human trajectory prediction," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6272–6281.

[3] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 2017, pp. 353–359.

[4] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 399–404.

[5] D.-D. Nguyen, C. Le Van, and M. I. Ali, "Vessel trajectory prediction using sequence-to-sequence models over spatial grid," in *Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems*, 2018, pp. 258–261.

[6] R. Jiang, X. Song, D. Huang, X. Song, T. Xia, Z. Cai, Z. Wang, K.-S. Kim, and R. Shibasaki, "Deepurbanevent: A system for predicting citywide crowd dynamics at big events," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2114–2122.

[7] R. Jiang, Z. Cai, Z. Wang, C. Yang, Z. Fan, Q. Chen, K. Tsubouchi, X. Song, and R. Shibasaki, "Deepcrowd: A deep model for large-scale citywide crowd density and flow prediction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 276–290, 2021.

[8] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[9] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, pp. 1–55, 2014.

[10] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[11] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 6120–6127.

[12] Z. Yan, "Traj-arima: A spatial-time series model for network-constrained trajectory," in *Proceedings of the Third International Workshop on Computational Transportation Science*, 2010, pp. 11–16.

[13] X. Zhang, X. Fu, Z. Xiao, H. Xu, and Z. Qin, "Vessel trajectory prediction in maritime transportation: Current approaches and beyond," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 19 980–19 998, 2022.

[14] L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle trajectory prediction using lstms with spatial–temporal attention mechanisms," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 2, pp. 197–208, 2022.

[15] Y. Xu, D. Ren, M. Li, Y. Chen, M. Fan, and H. Xia, "Tra2tra: Trajectory-to-trajectory prediction with a global social spatial-temporal attentive neural network," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1574–1581, 2021.

[16] D. Varshneya and G. Srinivasaraghavan, "Human trajectory prediction using spatially aware deep attention models," *arXiv preprint arXiv:1705.09436*, 2017.

[17] H. Xue, D. Q. Huynh, and M. Reynolds, "Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1186–1194.

[18] M. Huynh and G. Alaghband, "Trajectory prediction by coupling scene-lstm with human movement lstm," in *Advances in Visual Computing: 14th International Symposium on Visual Computing, ISVC 2019, Lake Tahoe, NV, USA, October 7–9, 2019, Proceedings, Part I 14*. Springer, 2019, pp. 244–259.

[19] G. Xie, A. Shangguan, R. Fei, W. Ji, W. Ma, and X. Hei, "Motion trajectory prediction based on a cnn-lstm sequential model," *Science China Information Sciences*, vol. 63, pp. 1–21, 2020.

[20] X. Zhang, X. Fu, Z. Xiao, H. Xu, and Z. Qin, "Vessel trajectory prediction in maritime transportation: Current approaches and beyond," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[21] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "Dnn-based prediction model for spatio-temporal data," in *Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2016, pp. 1–4.

[22] L. Wang, X. Geng, X. Ma, F. Liu, and Q. Yang, "Crowd flow prediction by deep spatio-temporal transfer learning," *arXiv preprint arXiv:1802.00386*, 2018.

[23] Z. Lin, J. Feng, Z. Lu, Y. Li, and D. Jin, "Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 1020–1027.

[24] S. Wang, J. Cao, H. Chen, H. Peng, and Z. Huang, "Seqst-gan: Seq2seq generative adversarial nets for multi-step urban crowd flow prediction," *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 6, no. 4, pp. 1–24, 2020.

[25] J. He, J. Wang, and Y. Luo, "Deep architectures for crowd flow prediction," in *Proceedings of the 2019 2nd International Conference on Data Science and Information Technology*, 2019, pp. 236–241.

[26] Q. Zhou, J.-J. Gu, C. Ling, W.-B. Li, Y. Zhuang, and J. Wang, "Exploiting multiple correlations among urban regions for crowd flow prediction," *Journal of Computer Science and Technology*, vol. 35, pp. 338–352, 2020.

[27] Z. Zou, P. Gao, and C. Yao, "City-level traffic flow prediction via lstm networks," in *Proceedings of the 2nd International Conference on Advances in Image Processing*, 2018, pp. 149–153.

[28] H. Yu, Q. Zheng, S. Qian, and Y. Zhang, "A fuzzy-based convolutional lstm network approach for citywide traffic flow prediction," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 3360–3367.

[29] G. Tang, B. Li, H.-N. Dai, and X. Zheng, "Sprnn: A spatial–temporal recurrent neural network for crowd flow prediction," *Information Sciences*, vol. 614, pp. 19–34, 2022.

[30] Q. Zhou, J. Gu, X. Lu, F. Zhuang, Y. Zhao, Q. Wang, and X. Zhang, "Modeling heterogeneous relations across multiple modes for potential crowd flow prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4723–4731.

[31] A. Ali, Y. Zhu, and M. Zakarya, "A data aggregation based approach to exploit dynamic spatio-temporal correlations for citywide crowd flows prediction in fog computing," *Multimedia Tools and Applications*, pp. 1–33, 2021.

[32] Z. Fan, X. Song, R. Shibasaki, and R. Adachi, "Citymomentum: an online approach for crowd behavior prediction at a citywide level," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015, pp. 559–569.

[33] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[34] S. S. Sohn, H. Zhou, S. Moon, S. Yoon, V. Pavlovic, and M. Kapadia, "Laying the foundations of deep long-term crowd flow prediction," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*. Springer, 2020, pp. 711–728.

[35] X. Wang, Z. Zhou, Y. Zhao, X. Zhang, K. Xing, F. Xiao, Z. Yang, and Y. Liu, "Improving urban crowd flow prediction on flexible region partition," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2804–2817, 2019.

[36] T. Zang, Y. Zhu, Y. Xu, and J. Yu, "Jointly modeling spatio–temporal dependencies and daily flow correlations for crowd flow prediction," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 4, pp. 1–20, 2021.

[37] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.

[38] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1059–1068.

[39] H. Ko, S. Lee, Y. Park, and A. Choi, "A survey of recommendation systems: recommendation models, techniques, and application fields," *Electronics*, vol. 11, no. 1, p. 141, 2022.

[40] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[41] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[42] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.