

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

2022 年度  
修 士 論 文

A Real-Time Trajectory Prediction System Based on the  
Meta-Learning Paradigm  
メタ学習パラダイムに基づくリアルタイム軌跡予測システム

2022 年 7 月 15 日提出  
指導教員 柴崎 亮介 教授  
副 指導教員 瀬崎 薫 教授

楊 曉傑  
Yang Xiaojie



# Abstract

Human mobility prediction is crucial for developing smart cities in the digital twin framework. As a digital replica of our world, the digital twin of the traffic system has been built from real-time data transmitted by GPS sensors attached to cars and has been leveraged to extract congestion and accidents. Today, as many people have become accustomed to carrying their smartphones, human mobility can be enriched by the location records of these smart devices and not only in-car navigation systems. The ever-increasing amount of data makes it possible to create a digital twin for a significant number of people within a metropolitan area. Nevertheless, real-time prediction relies on real-time data streams in the digital twin framework. In the development of smart cities, collecting data from people is one of the critical components for the prediction task. However, though previous research has developed many predictive models for human mobility, few of them have been modified into implement real-time systems. From an efficiency point of view, sometimes these models are not flexible enough in real-time systems because they are computationally intensive and time-consuming. Also, they lack mechanisms to help models distinguish differences between real-time and historical data. Based on the above issues and considerations, this research introduced the meta-learning paradigm to help predict people's movements with a GRU model at different times and built a retrieval-based historical trajectory database to fit the trajectory between the last observation and predicted outcomes. Finally, this research also proposed a real-time system structure and embedded the proposed model to evaluate the framework's performance under some special events using a simulated real-time data stream.

**Keywords:** human mobility, trajectory prediction, meta-learning paradigm, real-time system

# Contents

Section 1	Introduction	1
1.1	Background . . . . .	1
1.2	Related works . . . . .	4
1.3	Structure . . . . .	6
Section 2	Proposed methods	8
2.1	Preliminaries . . . . .	8
2.2	Methodology . . . . .	10
2.3	Baseline . . . . .	24
2.4	Evaluation methods . . . . .	24
Section 3	Experiments	26
3.1	Data description . . . . .	26
3.2	Experimental setups . . . . .	27
3.3	Evaluation on cluster-level prediction . . . . .	27
3.4	Evaluation on node-level prediction . . . . .	28
3.5	Evaluation on time efficiency . . . . .	30
3.6	Evaluation on real-time implementation . . . . .	33
Section 4	Conclusions and Prospects	35
4.1	Conclusions . . . . .	35
4.2	Prospects . . . . .	36
	Acknowledgement	38
	Reference	40



# List of Figures

1.1	Digital replicas of real-world entities . . . . .	2
1.2	Next-location Prediction . . . . .	3
1.3	Meta-learning paradigm . . . . .	6
2.1	Raw GPS record with UTC time format . . . . .	9
2.2	Interpolated trajectory (left), cluster-level trajectory (right) . . . . .	9
2.3	Most frequently visited hexagons (left), top 1601-3600 hexagons (middle) and sampled rest of hexagons (rights) . . . . .	11
2.4	Visit frequency of cluster-level locations in our study area . . . . .	12
2.5	Traditional GRU-based predictive model . . . . .	13
2.6	Pooling layer product: crowd-context of a day . . . . .	14
2.7	Proposed GRU-based predictive model . . . . .	16
2.8	Reduction in training loss of three GRU models . . . . .	16
2.9	Daily loss of the training data with a threshold band (orange) . . . . .	17
2.10	Illustration of making decision of railway/road trajectory with future infor- mation . . . . .	18
2.11	Extracting stay points from raw GPS data . . . . .	18
2.12	Dynamic interpolation: raw GPS data comes in and replace the formerly filled time slots based on algorithm 1 . . . . .	22
2.13	Structure design of proposed real-time prediction system . . . . .	23
3.1	Geospatial boundary of our research . . . . .	26
3.2	Daily Cross-Entropy losses of weekdays and weekends (x-axis refers to the time slot of the day) . . . . .	29
3.3	Gathering pattern of Tokyo station at morning rush hour . . . . .	30
3.4	Node-level prediction results evaluation . . . . .	31
3.5	Performance of designed real-time system . . . . .	34

# List of Tables

3.1	Evaluation on cluster-level Prediction . . . . .	28
3.2	Evaluation on Node-level Prediction . . . . .	32
3.3	Average time efficiency of different models . . . . .	33

# Section 1

## Introduction

### 1.1 Background

The rapid development of today's world has driven the era of big data, with billions of data being generated by people every day in a city. Among them, the mobile data has been considered a core component of creating smart transportation systems. Researchers have tended to mine historical movement data in an offline environment for reliable patterns and create models to verify how instructive these patterns are for future tendencies. However, this thinking also became a bottleneck in development due to explosive data volume growth and the consumption of complicated algorithms. To that end, the digital twin concept was proposed to meet this challenge. Illustrated in Fig.1.1, the digital twin is a digital replica of a real-world object in the virtual world, and they are synchronized by providing new data constantly[1]. In a smart transportation system, this concept is applied to detect the traffic flow in real-time to reduce congestion and accidents[2]. As data is updated, the digital twin of a smart city is always associated with a real-time state representation of the real-world entities, and changes in the state will reflect some objective pattern of the real world. More importantly, these patterns have a great potential to help us estimate the future state of the system.

As we mentioned, human mobility becomes more complex and diverse as the population in urban areas increases and the interaction between people and the environment becomes more pronounced. In metropolitan areas, human mobility patterns are constrained by the type of travel patterns and regional or wide-area events, and conversely, changes in the environment, such as noise and accidents, can affect people's propensity to travel. Therefore, the study of human mobility is crucial due to its essential role in many aspects of our society, such as urban planning[3], pollution management[4], and infectious disease control[5, 6], and

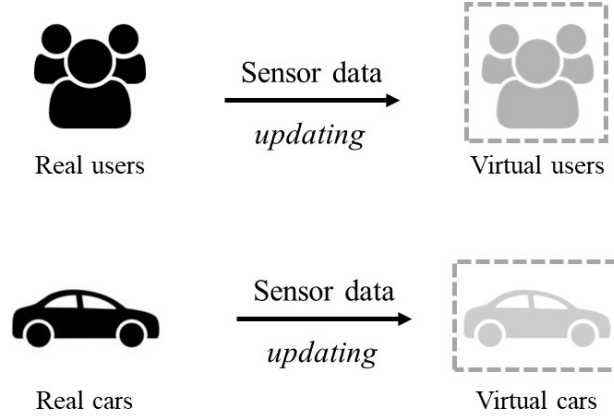


Fig.1.1: Digital replicas of real-world entities

natural disasters response[7]. For policymakers, regularity and irregularity can instruct them to change policies and laws to meet people's demands. The analysis of mobility patterns, in the end, is the key to human mobility-related research.

Previously, location data was typically generated and reported with specific devices attached to vehicles or ships, such as in-vehicle navigation systems. Currently, with the widespread development of smart devices, including mobile phones and the Internet of Things (IoT), more and more products can sense human mobility efficiently[8]. Today's most common positioning technology relies on global navigation satellite systems, among which GPS is the most commonly used system. This technology is mature and economical enough that almost all devices today are equipped with GPS chips, such as smartphones, self-driving cars, and drones. These devices, smartphones specifically, already have a very high market share. According to previous research, by the end of 2021, the number of smartphone users in the world would reach 3.8 billion[9]. At the same time, with the development of 5G communication technology accompanied by higher throughput, higher mobility, and higher transmission reliability with lower latency[10], smartphones will transmit internally generated sensor data, such as the location data we are interested in in this research, in more real-time. Therefore, on the scale of a metropolis, millions of GPS data will be generated every day to show people's movements every day[11], which provides the data support for building a digital twin from a human mobility perspective. Many previous studies focused on transportation mode classification, home-workplace detection, and population inference, with semantic information extracted from relatively small human mobility datasets[12]. Recently, the rapid development of computer science has made it possible to handle larger scale data, and the powerful Artificial Intelligence (AI) also makes it easier to analyze big

data of human mobility with complicated structures to complete predictive tasks[13].

Predictive tasks of human mobility contain two main categories, crowd flow prediction and next-location prediction[13]. The crowd flow prediction studies population movement flow on a fixed city scale. The research area will be divided into grids and attached with aggregated statistics like the number of people in the grid. The prediction will output the future population distribution in the research area by collecting history crowd flow data[14, 15, 16]. Additionally, the next-location prediction will focus on individual mobility. With a sequential set of independent human mobility data like GPS records, the prediction will capture patterns hidden in the sequential data to indicate human spatiotemporal regularity and finally output the future location with the information processed by the model[17, 18, 19]. Compared with aggregated human crowd flow data, the next-location prediction will reveal more information for its ability to analyze users separately. Meanwhile, knowing where people are heading will be more helpful in optimizing public resource allocation for policymakers, especially irregular events shown in Fig.1.2 including the O'bon festival, New Year's Eve, or more unpredictable events such as an earthquake or a tsunami[20]. Since analyzing individual-based movement patterns can be more specific and challenging, this study focuses on the next-location prediction.

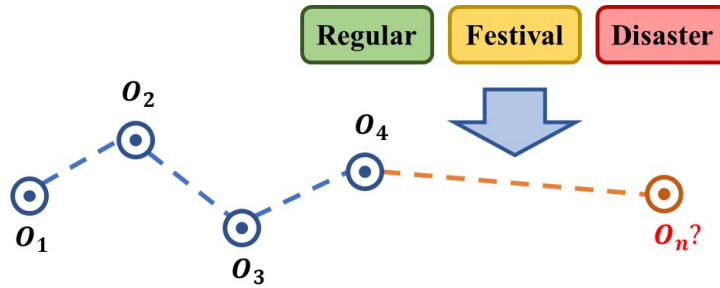


Fig.1.2: Next-location Prediction

Moreover, the human mobility data, specifically GPS records, is collected from users' devices distributed in the study area. Generally, it takes some time to process these raw data consisting of spatial drifting, delay, and other issues caused by data transmission[21]. For the development of the human mobility digital twin, we hope to process the data in a real-time system, just like a smart transportation system, where the previously mentioned issues may adversely affect the system's robustness. For some preliminary analysis, such as traffic flow visualization and congestion detection, it is always easy to realize them with real-time raw GPS data. However, when it comes to prediction, it is time-consuming for processes from raw data filtering to predictive model application. As a result, proper mechanisms and a rela-

tively simple model are necessary, to make the processing of human mobility and prediction smoothly run in a real-time system.

Considering the necessity and challenge of human mobility prediction in a real-time environment, we proposed a predictive model based on the meta-learning paradigm in this research. The model will be embedded in a simple framework to process simulated real-time raw GPS data stream to evaluate its performance in prediction accuracy and time efficiency.

## 1.2 Related works

Modeling human mobility mainly focuses on exploring the spatiotemporal characteristics and potential patterns hidden in individual and population trajectories[22]. With this consensus, city government can better theorize for future policy adjustments in urban management, such as infectious disease control[23], migration flow forecasting[24, 25] and urban planning[26]. Prediction work is promising because of the necessity of understanding the spatiotemporal semantic information in human mobility. Previous research has proved that human mobility has 93% potential predictability, and it always has a close relationship with social contexts, urban geography, and spatial constraints[27]. People always spend their time traveling among a few locations[28] instead of moving randomly or displacing for a long time[29]. This movement characteristic is recorded by people's trajectories, including GPS data or check-in data of Point of Interests (POIs).

Given a series of historical trajectories, the next-location prediction task requires us to understand hidden patterns and underlying features among them through both spatial and temporal perspectives. Before machine learning became mainstream, the Markov model was often used in prediction tasks for its high efficiency, simplicity, and low computational cost. For example, the Markov chain prediction model considered the last visited position of the position sequence to predict the user's next position[30], though such a stochastic model did not value the predict potential in trajectory information. Meanwhile, with the increasing performance of deep learning methods, more and more works have introduced deep learning models and related mechanisms. As each person has a significant probability of returning to a few highly frequented locations[28], human mobility exhibits strong spatiotemporal regularities, which can be leveraged for predicting urban human mobility [12]. With that in mind, previous research started to use historical trajectories as input for next-location prediction with deep learning models, as demonstrated in Fig.1.2.

Consequently, Recurrent Neural Network (RNN), a basic structure to deal with time-series predictive tasks, is widely used in next-location prediction[17, 31, 32, 33]. However, due to

the drawback that RNN cannot capture the long temporal dependency from input data[34], Song et al.[35] built a deep Long-Short Term Memory Learning (LSTM) architecture for urban human mobility prediction. Similarly, the Gated Recurrent Unit (GRU)[36] was also proposed to solve the same issue. Later studies have developed more sophisticated models and mechanisms to predict more accurately based on these deep learning models. Deepmove[17] designed a multi-model embedding RNN to capture sequential transitions in human mobility and utilize the periodical nature of history trajectories to augment mobility prediction. ST-RNN[18] fully used spatial and temporal information in trajectory data and built time-specific and distance-specific transition matrices for mobility prediction in different spatiotemporal situations. HST-LSTM[37] consisted of a hierarchical extension in the proposed spatiotemporal LSTM model in an encoder-decoder manner which modeled the contextual history visit information to boost the performance of the prediction model. Besides, the attention mechanism, a method to find the more substantial context from historical information in time series data, is widely applied to spatiotemporal predictive tasks. Flashback[19], combined with distance in spatial and temporal dimensions, was proposed by modeling sparse mobility traces and finding the hidden historical patterns with high predictive power for location prediction. VANext[31] introduced the attention mechanism by learning the weight of convolutional states in historical trajectories, while the attention methods were based on mobility patterns representation combining historical check-in data and variational query variables.

The mentioned models or mechanisms were generally designed for public datasets with limited providers and sparse data distribution, such as New York City Taxi Trip Data[17, 38, 39] and Foursquare check-in records[18, 19, 31]. None of the mentioned models is deployed into a real-time system for prediction. Predictably, due to the dynamicity of big data streams, high-frequency unlabeled data generation from the heterogeneous data sources, and complexity of human mobility[40], a real-time prediction algorithm is challenging and has more fluctuating instability[21]. We believe a real-time system should be robust to the newly received data instead of preprocessed specific datasets. The predictive model in the system should be dynamically self-adapted to the real-time environment and updated if irregularities happen, such as train delays, festivals, or even terrible disasters[41].

To that end, we naturally introduce the meta-learning paradigm, which can adapt the predictive model to different prediction tasks with corresponding prior knowledge. Meta-learning is incredibly prevalent in natural science, psychology, and neuroscience[42]. The Fig.1.3 shows the details of the paradigm in deep learning that the prior knowledge is constructed to solve the predictive tasks on different datasets (A and B), and the performance

of the prediction model will be improved on the target tasks[43]. In the supervised case, AI and deep learning leverage the paradigm to extract an appropriate prior of a task for error on this new task will be minimized[42]. The popular meta-learning method, model-agnostic meta-learning (MAML)[44], which consisted of a meta-learner and base-learner, was aimed to train a meta-learner to find a well-generalized initialization for base-learner. Previous research has already developed several structures for meta-learning, including MAML[44] and Reptile[45]. With such a method, MetaST[46] learned spatiotemporal patterns in traffic flow from multiple cities with rich data recourse for target cities with limited data for traffic flow prediction. Similarly, with sufficient data, memory-based approaches[17, 18, 37, 19] like RNN which also can be regarded as meta-learning to some degree, learn to find the proper weights so that the time-dependent indicators can effectively track the global state of the current task. Inspired by this concept, the proposal will investigate the predictive capability of memory-based models with designed meta-learning indicators.

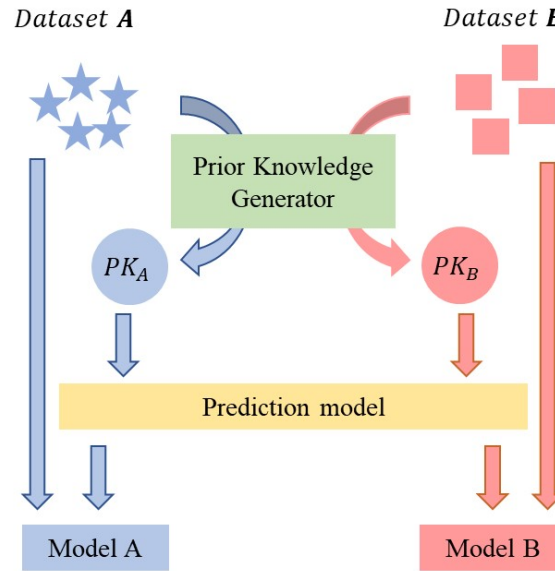


Fig.1.3: Meta-learning paradigm

### 1.3 Structure

In this research, we proposed a next-location prediction method from the perspective of the meta-learning paradigm. We investigated how the introduction of a meta-learning indicator (prior knowledge generator in Fig.1.3) would improve the performance of our proposed GRU predictive model. Then, we built a history trajectory retrieval mechanism to enrich the pre-



dictive results and implemented the predictive model into a real-time system with a simulated raw GPS data stream to evaluate our design system. Including this section, this paper has a total of 4 sections.

- Section 1 Introduce the background and related works of the research
- Section 2 Explain the proposed method and list the baselines in this research
- Section 3 Describe the experimental data and the performance of methods
- Section 4 Discuss the results of the experiment and prospects of the research

## Section 2

# Proposed methods

### 2.1 Preliminaries

In this section, we formulated the basic definitions of our proposed next-location prediction method and some following processes.

**Definition 1 (Raw GPS data)** As shown in Fig.2.1, the raw GPS data record can be formally described as follows:

$$X_{raw} = \{(user\_id, time, latitude, longitude)\} \quad (2.1)$$

Furthermore, to simplify data structure in the system, the raw data is organized as:

$$x_t^u = \{(latitude, longitude)\} \quad (2.2)$$

where  $u$  refers to  $user\_id$ , and  $t$  represents the time-slot index transformed from  $time$ . In this research, one day is divided into 288 time-slots (with a 5-minute time interval), and  $t$  can be calculated to infer its index of the day ( $0 \leq t < 288$ ).

**Definition 2 (Interpolated trajectory)** In the offline experiments, we collected daily raw GPS data from all the users. Through preprocessing, all records of the  $d$ -th day will be converted into a list of coordinates by the user's id with the same length. Based on the daily 288 time slots mentioned in Definition 1, the interpolated trajectory of each user  $u$  in the  $d$ -th day will be aligned with the format formulated as follows:

$$TR_d^u = x_{d,0}^u, x_{d,1}^u, \dots, x_{d,t}^u \dots x_{d,286}^u, x_{d,287}^u \quad (2.3)$$

With formulated format in Equation 2.3, we defined the complete dataset as  $TR$ .

**Definition 3 (Cluster-level trajectory)** We need to discrete spatially continuous coordinates in  $TR$  with an index-able manner to generate the input of the deep learning model.

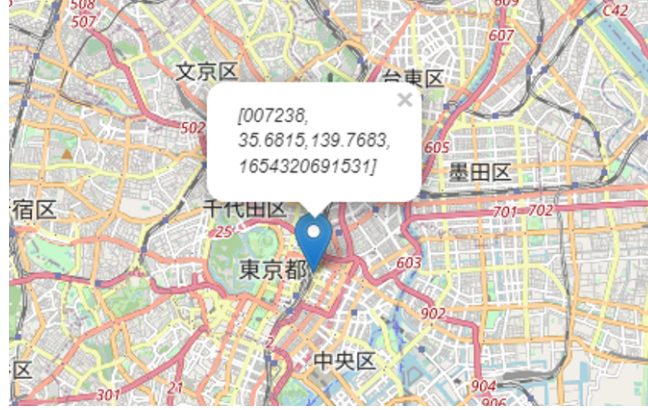


Fig.2.1: Raw GPS record with UTC time format

Indexed locations, also regarded as cluster-level locations, will be defined as  $\{c|c \in C\}$ . Once the user provides his location record  $X_{raw}$ , the coordinate (*latitude, longitude*) in  $X_{raw}$  will be converted to corresponding cluster index  $c$ . Transferred from interpolated trajectory  $TR_d^u$  illustrated in Fig.2.2, we have cluster-level trajectory  $TRC_d^u$  formulated as follows:

$$TRC_d^u = c_{d,0}^u, c_{d,1}^u, \dots c_{d,t}^u \dots c_{d,286}^u, c_{d,287}^u \quad (2.4)$$

All the cluster-level trajectories in our research will be defined as  $TRC$ , and in the section 2.2.1, we will explain how we chose clusters from the study area.



Fig.2.2: Interpolated trajectory (left), cluster-level trajectory (right)

**Definition 4 (Node-level trajectory)** We use preprocessed GPS data with the map-matching algorithm in offline experiments, and raw GPS data are allocated to the reasonably closest road network. Users' records are corrected to the road network with node index in the

road network and mesh code referencing Japan geocoding system. One single map-matched record can be presented as follows:

$$X_{node} = \{(user\_id, time, status, mode, latitude, longitude, node\_id, link\_id, meshcode)\} \quad (2.5)$$

where *status* refers to the moving state (STAY or MOVE) and *mode* refers to the transportation mode (WALK, BICYCLE, CAR, TRAIN OTHER) estimated for each user[19]. *node\_id* and *link\_id* are identification codes of nodes and edges in transportation networks, and *meshcode* is the grid number of the Japanese geocoding system, respectively. The detailed description will be stressed in 2.2.1.

## 2.2 Methodology

This section will discuss the methods and mechanisms for developing the real-time trajectory prediction system with a meta-learning paradigm with two main parts. The first part is the offline preparation, including model structure with meta-learning paradigm and retrieval-based historical trajectory database construction in section 2.2.1. Next, we will implement a simulated real-time environment with the proposed system design in section 2.2.2.

### 2.2.1 Offline preparation

Before developing the real-time system, we will design and examine the prediction model in an offline environment, with all the data prepared after noise filtering and map-matching. In this part, we intend to design a deep learning model with a meta-learning paradigm to predict next locations at the cluster level for all users with a good performance and explain how our proposed retrieval-based historical trajectory database works.

#### 2.2.1.1 Data preprocessing

In order to model patterns in trajectories from temporal and spatial dimensions, we need to discrete and align spatiotemporal information in raw GPS data and transform it into cluster-level trajectories, as we mentioned in Definition 3. We generated a cluster-level trajectory with the following six steps:

- Forwarding-filling the raw GPS data with a 15-minute interval for Null records after map-matching. It is worth noting that it is impossible to get forwarding-filled records before users send their data to the real-time system, while for offline preparation, we

have collected all the raw data in advance.

- Interpolating and sampling trajectory in a 5-minute interval after loading the data from the previous step into memory, for every user, we have a total of 288 records representing his movement for one day like Definition 2.
- Indexing the locations in trajectory using Uber’s Hexagonal Hierarchical Spatial Index (H3 index)<sup>\*1</sup> with the eighth resolution (average hexagon area  $0.74km^2$ ).
- Counting the visit frequency of the hexagons in the cluster-level trajectory dataset *TRC* in Definition 3, and creating two sets of hexagons: i) top 1600 most frequently visited hexagons, as illustrated in Fig. 2.3 (left), and ii) ranked 1601-3600 hexagons from the rest of hexagons with the probability proportional to their visit frequencies, as shown in Fig. 2.3 (middle).
- Simplifying the H3-indexed trajectory by approximating those H3-indices that do not belong to Type i) by the nearest Type ii) hexagon, which is illustrated in 2.3 (right). Thus, we use 3600 indices to represent all the locations in the dataset.
- Converting selected 3600 hexagons into the digital index from 0 to 3599, corresponding hexagons from the highest visit frequency to the lowest visit frequency illustrated in Fig.2.4. Currently, all the trajectories can be simplified as a series of indexes (we will still use *TRC* to present indexed cluster level trajectory).

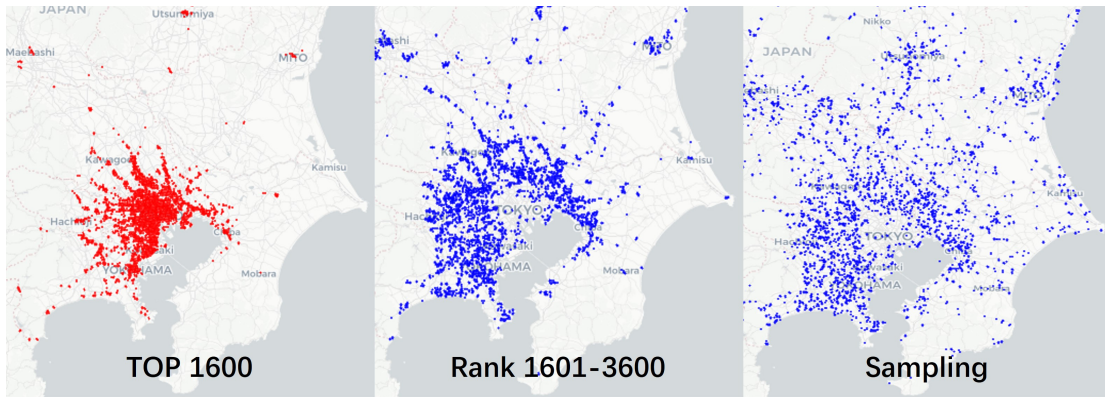


Fig.2.3: Most frequently visited hexagons (left), top 1601-3600 hexagons (middle) and sampled rest of hexagons (rights)

We leveraged the indexed cluster-level trajectory because the number of locations must be limited as the input of deep learning models. We introduced Uber’s Hexagonal Hierarchical Spatial Index to discretize continuous coordinate data. Nevertheless, as illustrated in Fig.2.4,

<sup>\*1</sup> <https://uber.github.io/h3/>

actually we have 38435 clusters in our study area, which is still too heavy for a predictive model. To reduce the burden of model training, we selected the top 1600 clusters (left part of the red line) and clusters with less visit frequency (the part between red and orange lines) which take 65.8% of all cluster-level locations, as subjects where we are taking attention. As mentioned in the listed steps and Fig.2.3, the rest clusters are fused into the top 1601-3600 clusters by distance to make our model more feasible to train.

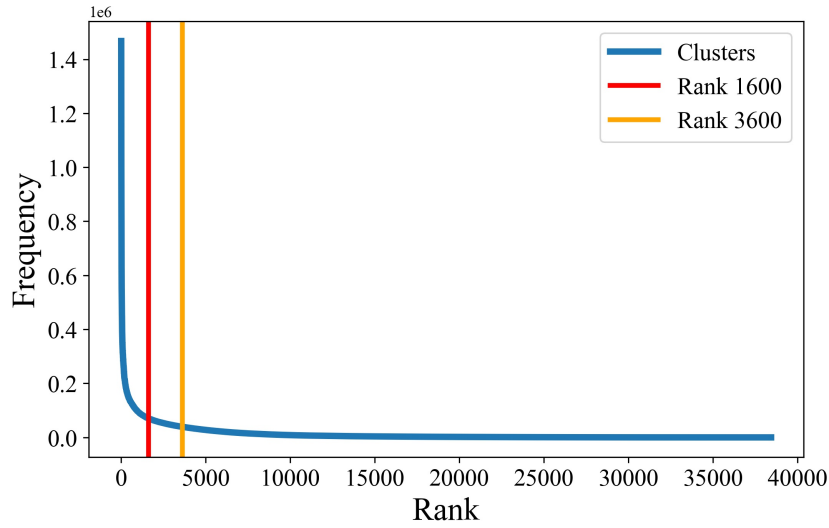


Fig.2.4: Visit frequency of cluster-level locations in our study area

### 2.2.1.2 GRU-structured deep learning model

Predictive models for human mobility generally consider historical information as input, based on that, our goal for the model is to predict the next location after a fixed time  $\Delta T$  with  $\Delta T$  historical trajectory of users, which can be declared as:

$$p\left(TRC_{t+2*\Delta T} \mid TRC_{t:t+\Delta T}\right) = f\left(TRC_{t:t+\Delta T}\right) \quad (2.6)$$

where  $f(x)$  is the predictive function that takes history trajectory in previous  $\Delta T$  time slots after time  $t + \Delta T$  as input, and creates a probability distribution on all the indexed clusters  $p$ , where the final destination will be selected. In this research, we introduced Gated Recurrent Unit (GRU) as this predictive function to model sequential data like cluster-level trajectory, which can be described as:

$$\begin{aligned} h_0 &= \mathbf{0} \\ h_\tau &= GRU([EL(TRC_{t+\tau}), ET(t)], h_{\tau-1}) \\ o_T &= SoftMax(MLP(h_{\Delta T})) \end{aligned} \quad (2.7)$$

In the traditional GRU we exemplified in the Equation 2.7 and Fig.2.5,  $h_0$  is the initialized hidden state in GRU (usually a zero tensor),  $h_\tau$  is the output of  $\tau$ -th recurrent.  $EL$  and  $ET$  are embedding layers for vectoring clustering index and temporal index.  $MLP$  represents multiple layers that we arranged to process the final output of GRU. The model will generate a vector  $o_T$  representing the probability distribution of moving to different clusters with *SoftMax* layer.

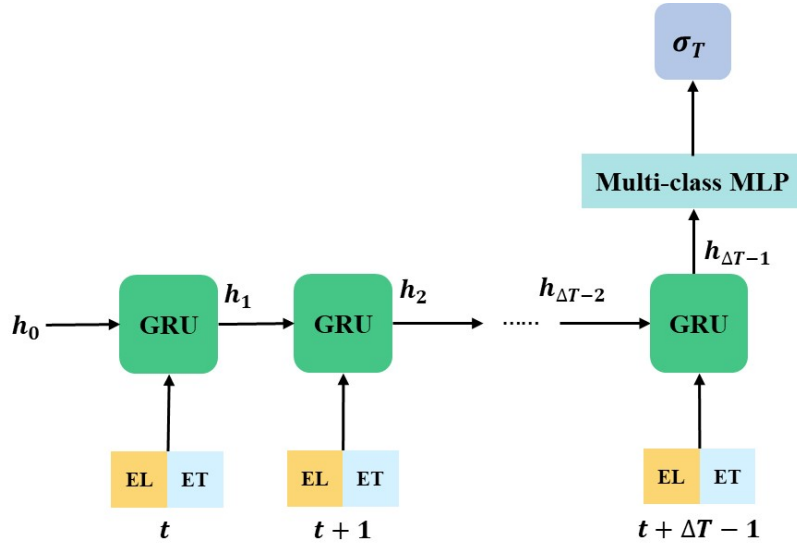


Fig.2.5: Traditional GRU-based predictive model

Based on what we introduced in Section 1.2, memory-based approaches like the RNN also embody the essence of the meta-learning paradigm[42]. As a variant of the RNN model, the GRU in Equation 2.7, is also trained with prior knowledge which is the hidden station  $h_{\tau-1}$  for the next state  $h_\tau$ . We believe the prior knowledge could be designed more wisely than the hidden state  $h_\tau$ . Considering the prediction tasks are organized by data with different levels of confusion in different time slots of a day, we decided to summarize the prior knowledge generator  $g(x)$  with input data  $TRC_{t:t+\Delta T}$  before prediction. As a result, the Equation 2.6 will be upgraded as follows:

$$p\left(TRC_{t+2*\Delta T} \middle| TRC_{t:t+\Delta T}\right) = f\left(TRC_{t:t+\Delta T}, prior\right) \quad (2.8)$$

$prior = g\left(TRC_{t:t+\Delta T}\right)$

To design the prior knowledge generator  $g(x)$  for meta-learning paradigm, we need to understand and analyze the input data of predictive model  $TRC_{t:t+\Delta T}$ . The mentioned GRU structure of the Equation 2.7 could be applied to a single trajectory  $TRC_{t:t+\Delta T}^u$  to extract the spatiotemporal pattern for user  $u$ . However, when it comes to a larger-scale trajectory predic-



tion, we must consider the cross-interference between trajectories. More specifically, if one cluster has increasing visit frequency over time, we can believe that more and more users will be more likely to visit this cluster in the future. That means we must consider the collective pattern while deciding the destinations. The collective pattern could act as prior knowledge, which is flexible based on the current trajectories and help make a more dynamic decision for different events. As a solution, we appended a *Pooling* layer to calculate the crowd context  $\Phi$ , which aggregates with historical cluster level trajectory  $TRC_{t:t+\Delta T}$  of all the users to get a global state. The original GRU model is combined with the crowd context as prior knowledge as the Equation 2.9:

$$\begin{aligned}
 h_0 &= \mathbf{0} \\
 h_\tau &= GRU([EL(Trc[t + \tau]), ET(t + \tau)], h_{\tau-1}) \\
 \Phi_{\Delta T} &= Pooling(h_{\Delta T}) \\
 o_T &= SoftMax(MLP(h_{\Delta T}), \Phi_{\Delta T})
 \end{aligned} \tag{2.9}$$

As exhibited in Fig.2.6, the crowd-context keeps altering over time, presenting its ability to distinguish all users' daily global patterns.

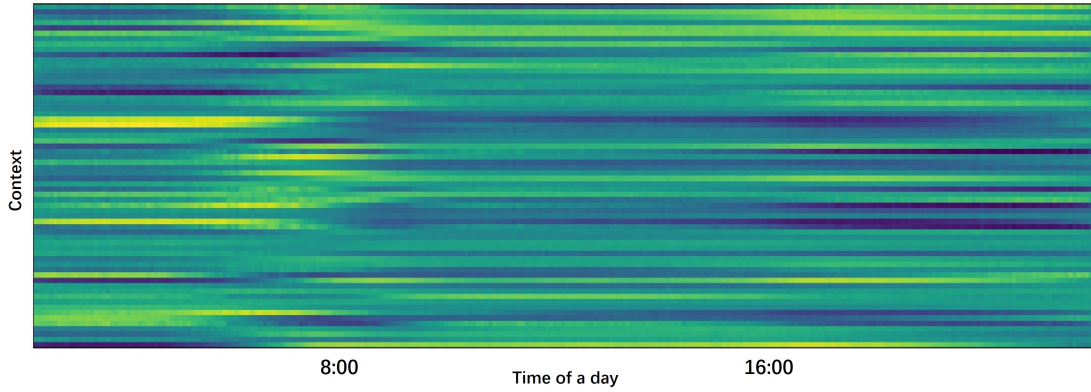


Fig.2.6: Pooling layer product: crowd-context of a day

The upgraded structure in the Equation 2.9 outputs the final hidden state at time-slot  $t + \Delta T$  combined with crowd context as prior knowledge, before being passed to the multiple-layer perceptron of *MLP*. Nevertheless, it ignores the spatiotemporal information in the coordinate-level GPS data *TR*. According to the Flashback attention mechanism[19], the hidden state for every iteration could consist of potential predictive power, represented by a designed context calculated from spatiotemporal differences. So we introduced this attention mechanism by fusing the personal context to the predictive results before the *SoftMax* layer. As the temporal difference in *TR* is the same for neighbor coordinates after interpolation



preprocessing, so we only considered attention weights caused by spatial displacement in the input data  $TRC_{t-\Delta T:t}$  showing as follows:

$$\begin{aligned} \Delta D_{t+\Delta, t+\tau} &= |TR_{t+\Delta} - Tr_{t+\tau}| \\ w_S(\Delta D) &= e^{-\beta \Delta D} \end{aligned} \quad (2.10)$$

where  $\Delta D$  is illustrated as L2 distance from the origin location at time slot  $t + \tau$  to time slot  $t + \Delta T$ ,  $\beta$  means a spatial decay rate,  $W_{t+\tau}$  refers to the context-aware weight for hidden state  $h_\tau$ .

Finally, to comprehensively make full use of the information in all the hidden states instead of the last one, we merged them into  $H_{\Delta T}$  before transiting them to the final *MLP* layer:

$$H_{\Delta T} = [h_0, h_1, \dots, h_\tau, \dots, h_{\Delta T}] \quad (2.11)$$

In summary, in order to make the predictive model be able to express more spatiotemporal information and have a better performance, the updated GRU-structure model is shown as follows and Fig.2.7:

$$\begin{aligned} h_0 &= \mathbf{0} \\ h_\tau &= w_{t+\tau:t+\Delta T} * GRU([EL(TRC_{t+\tau}), ET(t), EW(d)], h_{\tau-1}) \\ \Phi_{\Delta T} &= Pooling(h_{\Delta T}) \\ o_T &= SoftMax(MLP(H_{\Delta T}), \Phi_{\Delta T}) \end{aligned} \quad (2.12)$$

where  $\Phi_t$  represents the crowd context at prediction time  $t + \Delta T$ , this is an essential component for modeling large-scale spatial prediction on a city scale, as we mentioned before.  $EW(d)$  is a newly added embedding layer about the day.

In the end, we have three continuously optimized models of Equations 2.7, 2.9 and 2.12, for each we call them *GRU*, *GRU\_crowd\_context* and *proposed\_GRU(Ours)* respectively. They are assigned with the ability to learn from the memory-based structure and designed crowd context (for the last two methods) to generate prior knowledge for prediction, learning from the meta-learning paradigm. We evaluated the performance of three model structures with a degree of reduction in training loss illustrated in Fig.2.8. In this figure, we concluded that prior knowledge with more information benefit the training process of the deep learning model under the meta-learning paradigm.

### 2.2.1.3 Real-time adaptation

In this section, we will discuss the mechanism to make the deep learning model trained with a meta-learning paradigm able to self-adapt to irregular events. In this research, we want our proposed predictive model to be scalable to the potential events with prior knowledge  $\Phi$  in the

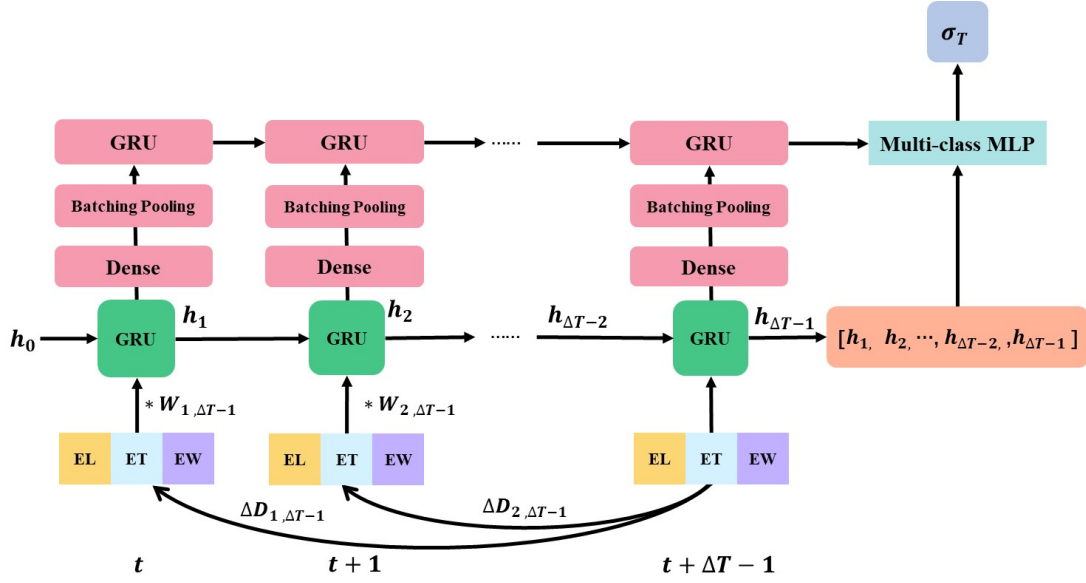


Fig.2.7: Proposed GRU-based predictive model

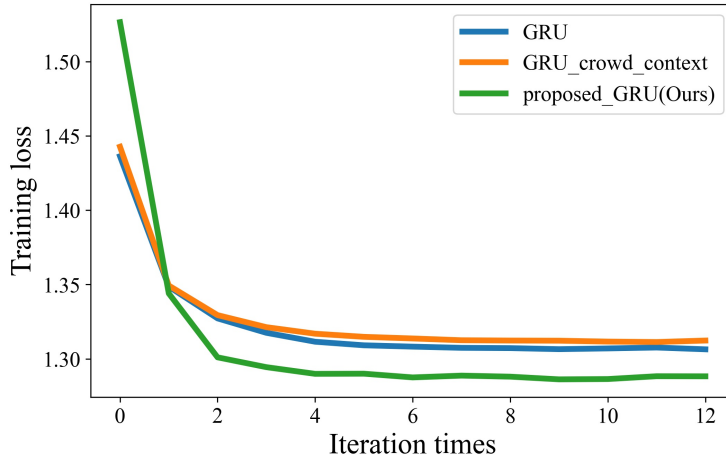


Fig.2.8: Reduction in training loss of three GRU models

Equation 2.9 and 2.12. Generally speaking, in our research, we can predict cluster-level next-location  $TRC_{t+2*\Delta T}$  with  $TRC_{t:t+\Delta T}$  in our offline experiments with known mobility patterns in our training datasets, and the *proposed\_GRU* performed well to distinguish existing events or patterns suggested in Fig.2.8.

Nevertheless, for our real-time implementation, the mobility pattern of ground truth  $TRC_{t+\Delta T:t+2*\Delta T}$  organized by raw GPS data may not be the same as the historical patterns. Considering that, we need to detect the irregular events for the newly collected data and modify the pre-trained model with collected  $TRC_{t:t+\Delta T}$  at time slot  $t + \Delta T$ .

We tried to use the cross-entropy loss  $loss_{t+\Delta t}$  of the model with input data  $TRC_{t-\Delta T:t}$  and

ground truth  $TRC_{t+\Delta T}$  at time  $t + \Delta T$ . We learned the regular loss of  $loss_t$  from training data and set a threshold for real-time implementation illustrated in Fig.2.9 ( $\pm 0.15$ ). If the performance of predictive model results in an outlier like red dot in Fig.2.9, we will dynamically modify the base-learner with  $TRC_{t:t+\Delta T}$ .

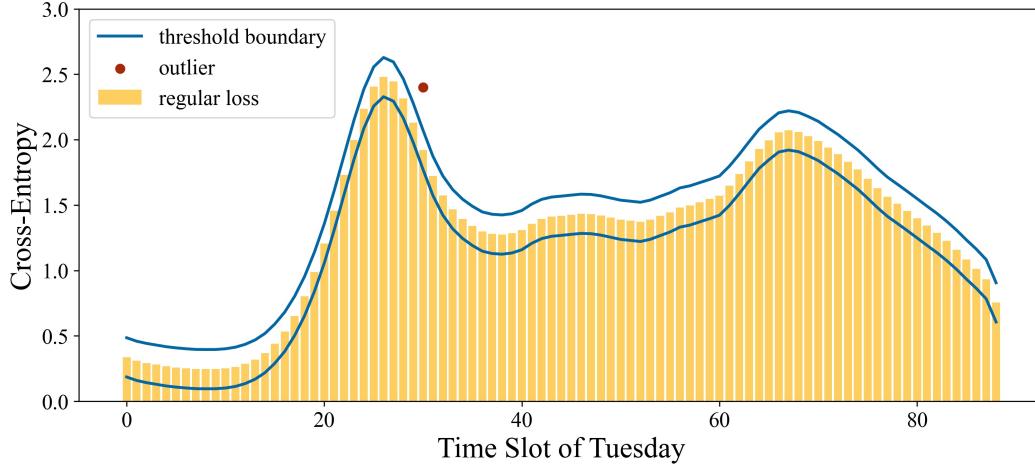


Fig.2.9: Daily loss of the training data with a threshold band (orange)

#### 2.2.1.4 Retrieval-based historical trajectory database

To model human mobility with trajectories, we need trajectory interpolation and map-matching to complement and enhance the representation of trajectories on transportation networks with coordinates in raw GPS data. Generally, map-matching is the best way to find the trajectory between two spatial locations. However, map-matching is time-consuming and relies on future information to eliminate uncertainties. As shown in Fig.2.10, we already know three coordinates (black icon) and generate a trajectory with them, but we cannot decide if the trajectory on the transportation networks as long as we receive the following GPS records. The algorithm will determine the trajectory on the road for the green icon, while the orange icon may increase the possibility of generating the trajectory along the railway. As a result, it is impractical to implement a map-matching algorithm into an real-time system where we do not know the user's next GPS-level location.

To this end, we proposed a retrieval-based historical trajectory database mechanism to fully use preprocessed map-matching trajectories, indicating the user's preference on path choice at any time in a day or a week.

As mentioned in definition 4, we conducted a three-step process to obtain a node-level trajectory with historical data, as shown in Fig.2.10.



Fig.2.10: Illustration of making decision of railway/road trajectory with future information

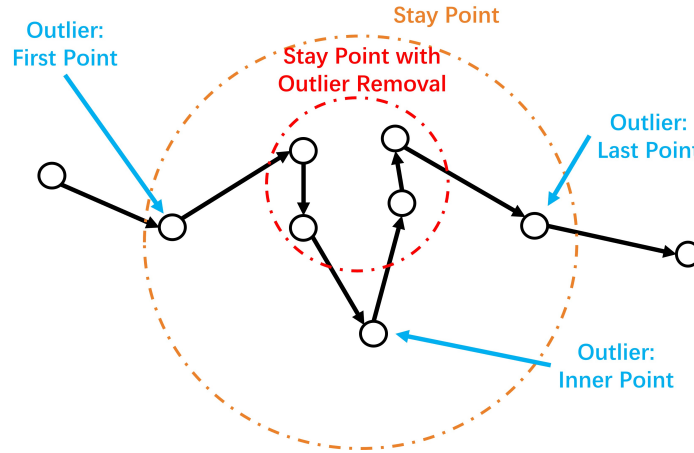


Fig.2.11: Extracting stay points from raw GPS data

1) Trip segmentation: from historical raw GPS data, stay points are decided with the 300-meter and the 15-minute spatiotemporal threshold, as shown in Fig.2.11. Furthermore, moving state (STAY or MOVE) and transportation mode (WALK, BICYCLE, CAR, TRAIN OTHER) are determined and partitioned for each trip based on the methods in [19].

2) Filling blank among segmented trips: the trips are defined based on GPS logs, and thus there are gaps between trips because of the lack of GPS data. Considering the STAY/MOVE states and spatiotemporal gaps (200 meters and 1 hour) among trips, we filled the blank between trips with the following rules:

- i) STAY  $\rightarrow$  STAY: If a long-distance gap exists between two trips, a MOVE state will be inserted into the mid-term. Otherwise, STAY will fill the mid-term instead.
- ii) STAY  $\rightarrow$  MOVE / MOVE  $\rightarrow$  STAY: If a state transition is detected between two trips, we will determine the transition time in the mid-term and insert the MOVE/STAY states to connect two trips.
- iii) MOVE  $\rightarrow$  MOVE: Merge the two consecutive MOVE trips if spatiotemporal gaps are greater than thresholds. Otherwise, estimate the potential STAY state in the midterm after extending the known MOVE states.

3) Road estimation: to transfer GPS points into trajectories on transportation networks, we combined the map-matching and route search to determine the more accurate travel time for every MOVE trip based on the transportation mode and road/railway type.

After generating map-matched node-level trajectories, the mechanism of is built with two main parts, the trajectory database construction, and the database index generation. The basic idea of the mechanism is to find the most similar trajectory feature compressed by spatiotemporal information as  $feat_t$  with predicted results in the same cluster-level destination. The detailed definition is as follows:

$$feat_t = [latitude_t, longitude_t, w \cdot \cos(2\pi \frac{t}{T}), w \cdot \sin(2\pi \frac{t}{T})] \quad (2.13)$$

where  $w$  is the temporal weight to increase the difference in the same coordinates at different times so that we can retrieve trajectory by location and time information.

We will explain the two mentioned parts in more detail as follows:

- Trajectory database construction. After collecting all the map-matched GPS node-level trajectory data, the process will cut every user's trajectory with a certain time interval, such as  $\Delta T$  (15 minutes in our experiments), so the trajectory data in one slice can be represented as:

$$Traj_t^u = \{X_{node} | uid = u, time \in [t, t + \Delta T)\} \quad (2.14)$$

After that, the historical trajectory database will be generated by  $key = uid + date + timeslot$  for every  $Traj_t^u$ , and the value: the trajectory information of  $X_{node}$  without the user's id and time columns.

- Database index generation. This part has two products, trajectory feature, and database index. For instance, trajectory feature of user  $u$  will extract information from cluster-level trajectories  $TRC_{t,t+\Delta T}^u$ , where cluster-level origin and destination are  $ori = TRC_t^u$  and  $dst = TRC_{t+\Delta T}^u$  respectively, then trajectory feature will be defined as  $feat_t^u$  with

the corresponding coordinate data, finally the database index will be defined as  $key_t^u$  mentioned in the former part. Next, we classified trajectory feature and database index based on origin-destination(OD) pairs  $[ori, dst]$ . After that, we will simplify both products with a  $n$ -length list where the  $n$  means the total number of clusters. For  $dst$ -th element in the lists  $traj\_feat$  and  $traj\_indx$  will be insert with trajectory feature and database index with OD of destination  $dst$ .

The proposed database can help us retrieve historical trajectories once we get the predictive cluster-level results. When we have predictive results  $\hat{dst}^u = T\hat{RC}_{t+2*\Delta T}^u$  with the last observed coordinate for user  $u$   $TR_{t+\Delta T-1}^u$ , we can generate a predicted trajectory feature as  $\hat{feat}_t^u$ . There are multiple features in the  $dst^u$ -th element of the list  $traj\_feat$ , and we need to find the most similar trajectory feature with  $\hat{feat}_t^u$  by Euclidean distances. Next, the trajectory index  $key_t^u$  will be found in the same location as the selected feature. Finally, we can query one historical trajectory from the historical trajectory database with the index  $key_t^u$ .

### 2.2.2 Real-time prediction system

The raw GPS data is generally collected by mobile phone carriers and then pushed into an real-time data stream. So, we need to design a structure to receive and process them in advance if we want to implement our proposed predictive model. Before that, we need to emphasize one of the characteristics of raw GPS data that may be unfavorable to our proposed real-time system structure: the late arrival data issue caused by delays associated with wireless signal transmission. For offline experiments, it is easy to assume that the GPS record with timestamp  $ts$  is received at time  $ts$ . However for an real-time environment, it's hard to get the record at that exact time, it's normal to receive data with timestamp  $ts$  with a certain delay  $\widetilde{\Delta ts}$  at  $ts + \widetilde{\Delta ts}$ .

We also supposed that one user's id  $u$  firstly appears in the user id set. In that case, we may fail to collect his historical cluster-level trajectory  $TRC_{t:t+\Delta T}^u$  at time  $t + \Delta T$ , which is empty values in our collected and processed data. We will interpolate this user's trajectory before training the predictive model during offline preprocessing. However, it may take some time for a real-time environment as we need to judge which record to expend to those time slots with empty values. Moreover, if we do not keep one's record for a longer time, users' last observed location in the evening, for example, will not be considered for the morning prediction, where we can highly believe that he is probably at his home.

We need to consider solutions to the above two issues before we embed the predictive

model into the real-time system. Otherwise, we may conduct an unstable system because of the uncertainty of real-time raw data. So we designed a method of dynamical interpolation based on received real-time raw data. The tailored designed dynamical interpolation method is described as follows with new raw GPS raw data  $X_{raw}$ :

---

**Algorithm 1:** Dynamic interpolation
 

---

```

1 Initialization  $TR, TRC$ 
2 while  $X_{raw}$  exists do
3    $x_t^u \leftarrow X_{raw}, c \leftarrow x_t^u$ 
4   if  $u$  in  $TR$  and  $TRC$  then
5     if  $t + \Delta T < n$  then
6       #timestamp is located at the end and the beginning of the day
7        $Idx = [t + \Delta T : n] + [0 : t]$ 
8     else
9       #timestamp is located at general hours
10       $Idx = [n - (t + \Delta T) : t]$ 
11    end
12     $TR_{Idx}^u = x_t^u, TRC_{Idx}^u = c$ 
13  end
14 end

```

---

In the Algorithm 1, the  $Idx$  in line 7 refers to the data in  $TR^u$  and  $TRC^u$  that need to be updated according to the  $t$ , the index of time slot of  $x_t^u$ . With this algorithm, the real-time system can automatically interpolate a new user's trajectory by coordinates and cluster-level index into daily  $n$  length records in  $TR$  and  $TRC$ . Moreover, it can update the existing user  $u$ 's trajectory by keeping the last  $\Delta T$  records unchanged. Moreover, the new record will place the rest of the records representing  $n - \Delta T$  time in the future (23 hours from time  $T + \Delta T$  in our experiments), as shown in Fig.2.12. As a result, we can extract the last  $\Delta T$  ground truth from  $TR$  and  $TRC$  at  $t + \Delta T$  of a day when a predictive process is scheduled in the real-time system. Besides, this algorithm can ensure that the last observations of inactive users who do not send their data can join in the prediction process along with other active users. This function will be beneficial for prediction in the morning because some users have "disappeared" since midnight, probably at home. Finally, in the case of those users who just passed by our study area by leaving several records, we should delete them when we found

their last record was over a long time, such as 24 hours. Otherwise, we may find too many not-moving users at some main train stations or the boundary of our study area.

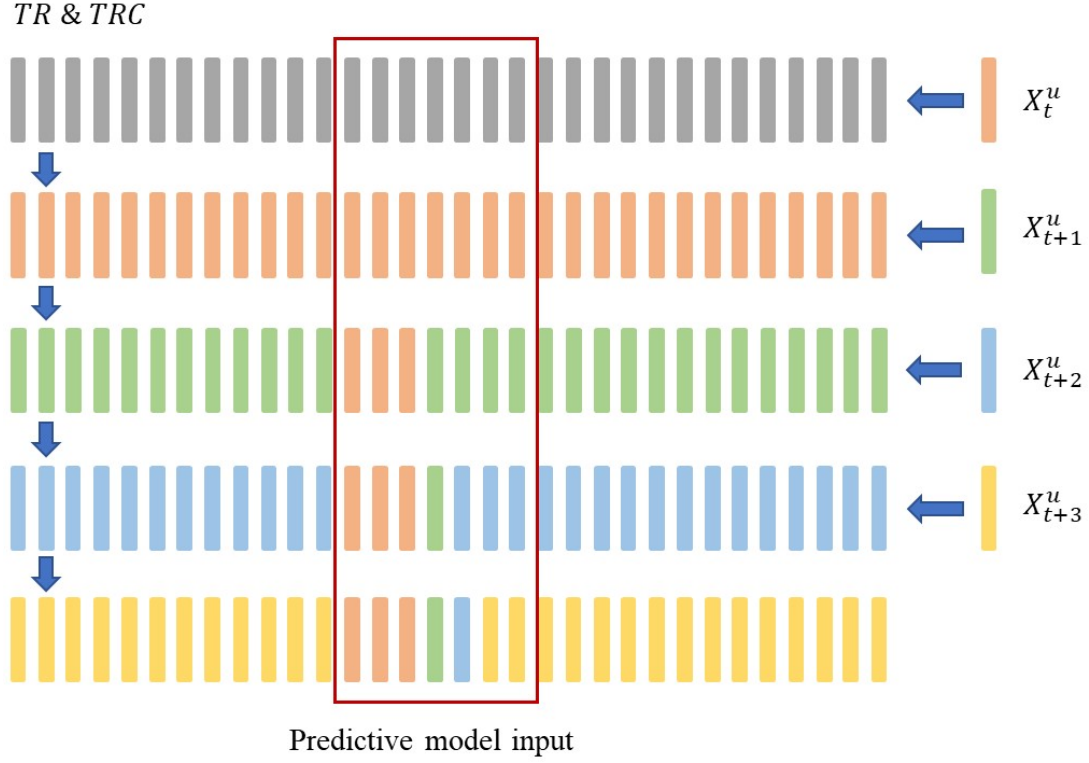


Fig.2.12: Dynamic interpolation: raw GPS data comes in and replace the formerly filled time slots based on algorithm 1

With the proposed algorithm 1, we designed a simple real-time system structure to deal with the real-time raw GPS data stream and predict users' next location along with the historical trajectory retrieval mechanism. We will discuss the structure in the following steps:

1. Record collector. This module will keep reading raw GPS data from the real-time data stream without stopping and will pass the new records to the next module because of a limitation of 5000 GPS records. In this step, one thing that needs to be emphasized is that some user may send their record too frequently in seconds, so we need to set a temporal threshold to make sure one single user-id will not take a too large proportion in the set of records.
2. Record processor. In the module, the system will receive the records from the previous step and apply dynamic interpolation in Algorithm 1 to update the user's interpolated trajectory data  $TR$  and cluster-level trajectory  $TRC$ , and finally save two data sets if the scheduled prediction is detected. Generally, we will monitor the time and start



prediction in every  $\Delta T/4$  (15 minutes in our experiments) time interval.

3. Prediction module. This module will load the saved two historical interpolated and cluster-level trajectories generated in the previous step. The system will extract data in the latest  $\Delta T$  time range as  $TR_{[t:t+\Delta T]}$  and  $TRC_{t:t+\Delta T}$  as input of predictive model pre-trained as Equation 2.7 described, eventually obtain the cluster-level predictive destination  $\hat{TRC}_{t+2\Delta T}$  for every user in data sets.
4. Historical trajectory retrieval. This step will utilize the database mentioned in 2.2.1. With predictive results  $\hat{TRC}_{t+2\Delta T}$  and the last observed records  $TR_{t+\Delta T}$ , the system will generate trajectory features for every user. Finally, we can extract the corresponding historical trajectory from the database after finding a similar historical trajectory feature.

With the proposed real-time system design, we can process real-time raw GPS data streams and predict users' next location in a relatively short time. We will discuss the efficiency and performance of the design in the next section.

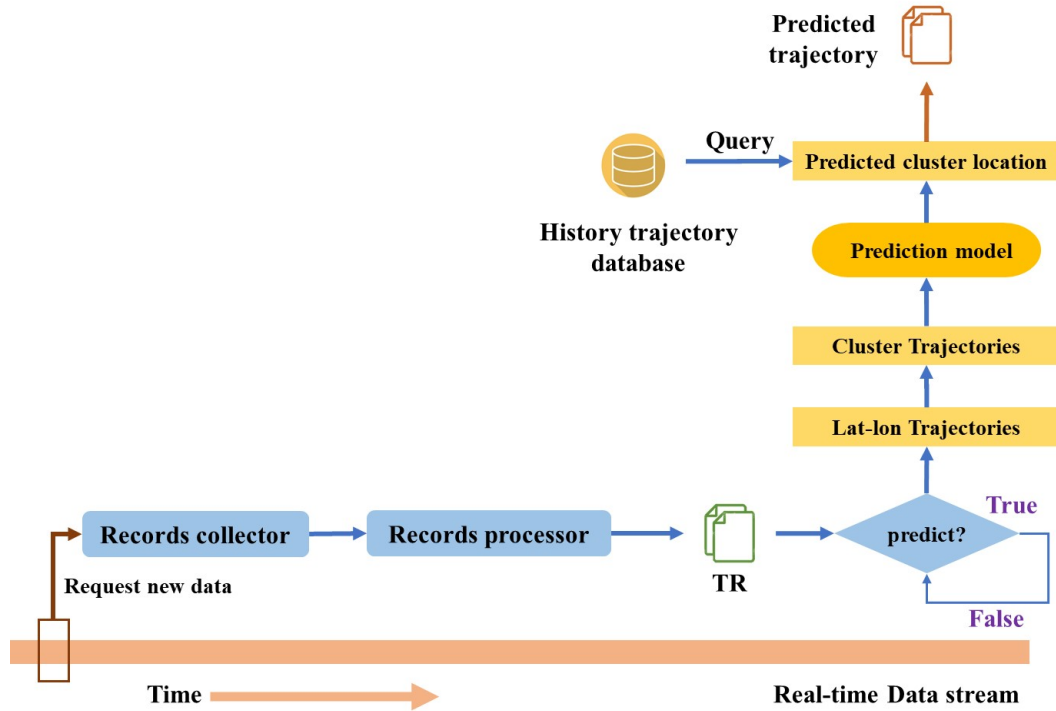


Fig.2.13: Structure design of proposed real-time prediction system

## 2.3 Baseline

This section will introduce baseline methodologies compared with our proposed predictive model. We selected two models designed as intermediate products during our research, one famous meta-learning framework and one state-of-art for next-location prediction. We hope to prove that our proposed model and structure have a good performance for regular patterns and can self-adapt to the latest data when some irregularities happen for the real-time system implementation. The baselines are listed and described as follows:

- **GRU**: This is an RNN model generally designed for time-series data prediction. We believe this is a basic deep learning model under the meta-learning paradigm for its memory-based hidden state structure. We will not consider spatiotemporal differences in this baseline, but the location indexes and time information as Equation 2.7.
- **GRU\_crowd\_context**: We upgrade the meta-learning prior knowledge function in this model by introducing the crowd context  $\Phi$  as Equation 2.9. The hidden state can improve the performance of the training, but the meta-learning indicator  $\Phi$  can distinguish the global states changing along with the time, compared with **GRU**.
- **MAML**: MAML is a famous meta-learning framework for few-shot learning of images. We applied the framework as a meta-learner and embedded the proposed model as a base-learner in the MAML. For every round of prediction, we will feed the MAML with a small scale of data to help the meta-learner know how to adapt the base-learner to the patterns in the new come-in data.
- **Deepmove**: This famous next-location prediction method proposed a historical attention model to capture the periodicity nature to augment the prediction ability of a next-location prediction RNN model. For our implementation, we applied Deepmove to users who share the same id among the whole dataset and GRU\_base to the other users for predictive tasks.

## 2.4 Evaluation methods

In this section, we will introduce our method of validation. For cluster-level next-location prediction, we utilized common strategies to evaluate our results, such as accuracy and recall rate (recall@5, recall@20) for offline and real-time environments.

Besides, we also tested the performance of our retrieval-based historical trajectory mecha-

nism. For our prediction, users will get one predicted future trajectory composed of a series of records  $X_{raw}$  in node-level trajectory. So we evaluated the node-level trajectory results with map-matched raw GPS data by counting the number of  $node\_id \in X_{raw}$  representing the visit volume to the referring important locations, such as train stations.

As we want to implement our proposed model into a real-time system, we tested the time efficiency of all the models. Finally, we created a simulated real-time data stream for our designed system and test some irregular events to evaluate the performance of our proposed model and the update mechanism.

## Section 3

# Experiments

### 3.1 Data description

In this research, we used a national dataset "Konzatsu-Tokei(R)" generated by individual location data sent from smartphones with enabled AUTO-GPS function under user consent through the "Docomo Map Navi" service provided by NTT DOCOMO, INC. Before we pre-processed these data, personal information was removed through the collective and statistics process considering the privacy issues. We used one month of data (from 2011.02.01 to 2011.02.28) for training/validating the proposed methodology in the previous section and 12 days of data (from 2011.03.01 to 2011.03.12) for evaluation. We filtered the data in the Kanto region with a prepared shapefile geospatial boundary in Fig.3.1 to help keep the experiment to a relatively small area.



Fig.3.1: Geospatial boundary of our research

## 3.2 Experimental setups

We deployed our algorithm on a deep learning workstation with Intel Xeon E5-2690v4 (2.6GHz 14C 35M 9.60 GT/sec 135W), 2 x TitanX Pascal 12GB GDDR5X, 128GB (8x 16GB DDR4-2400 ECC RDIMM) and 1.2TB Intel® NVMe SSD DC P3600 Series. The algorithm was implemented with Python 3.7.10 except for the trajectory interpolation and map-matching part with Java. We utilized the deep learning framework PyTorch 1.6.0 [47] to construct the cluster-level predictor and key-value storage library levelDB [48] (with python API plyvel 1.3.0) for building and online querying the fine-grained history trajectory database.

We set the time interval to be 5 minutes for trajectory interpolation, and we used the most recent one-hour (12-time slots) trajectories to make a one-hour ahead prediction every 15 minutes. Following the Equation 2.12, the embedding dimension of the cluster (EL), time (ET), and day (EW) was set to 128, 64, and 64, respectively, and a two-layered GRU with a hidden size of 256 was utilized to model the sequential pattern for individual trajectories, and a single-layered GRU with a hidden size of 64 is appended to model the sequential pattern of crowd context. The spatial decay parameter  $\beta$  of the equation 2.10 is set as 1000 to deal with coordinate differences. Multi-class MLP layer has a two-layered network with a 512 dimension latent layer.

## 3.3 Evaluation on cluster-level prediction

As we mentioned in Section 2.3 and 2.4, for the cluster-level prediction, we evaluated the performance of proposed model and baselines with recall rate, accuracy and cross-entropy, we will discuss their differences in the following parts:

- **Accuracy:** refers to the percentage of all correctly predicted clusters over all predicted outcomes;
- **Recall@k:** refers to the percentage of the number of collected results in all top  $k$  predicted clusters overall predicted outcomes;
- **Cross-entropy:** refers to the weighted sum of the difference between the distribution of each predicted cluster and the actual category.

After applying different models, based on proposed evaluation methods, we had the evaluation results of cluster-level prediction performance as Table.3.1 shows.

In the Table.3.1, we found that the baseline Deepmove performed much better even than our

Table3.1: Evaluation on cluster-level Prediction

	Cross_entropy	Accuracy	Recall@5	Recall@20
GRU	1.3463	0.6642	0.7878	0.8215
GRU_crowd_context	1.3530	0.6611	0.7873	0.8212
MAML	1.4773	0.6498	0.7841	0.8105
Deepmove	<b>0.9442</b>	<b>0.7754</b>	<b>0.8597</b>	<b>0.8882</b>
Ours	1.3350	0.6695	0.7870	0.8241

proposed method, that is because this model successfully extracted user's mobility patterns from their historical trajectories and had a preference for cluster-level prediction for a user following a daily routine, such as from home to workplace, and this kind of regular users make up the majority of the user set. However, we will discuss its disadvantages in the following sections.

Next, we found that the performance of our proposed model varied with the time slot of the day. In Fig.3.2 we illustrated the change of cross-entropy (here our proposed method is illustrated), which is the most preventative indicator that suggests the final cluster-level outputs fit the ground truth or not. Also, as shown in Fig.3.2 we found that the patterns of cross-entropy differ with weekdays or weekends. We used these results as our reference for real-time system to decide if we need to modified our predictive model.

### 3.4 Evaluation on node-level prediction

As illustrated in Fig.3.3, we visualized the predicted trajectories at a busy time, 07:00 on 2011.03.01 of Tuesday, a normal weekday morning, to see how did users gathering at some important places such as Tokyo station in the morning rush hour. One-hour ahead historical retrieved trajectories at prediction for 07:00 were compared with those trajectories passing Tokyo station during 07:45 - 08:00. We took a snapshot every 20 minutes and found our proposed method can predict trajectories passing to or through the Tokyo station quite well (e.g., where users start from and which route they take).

In Fig.3.4, we selected two types of aggregated analysis: 1) the number of passengers using the station on particular lines like Yamanote Line and Joban Line, and 2) the total number of passengers for all lines in an interchange, all the results are focused on two large train stations in Tokyo: Tokyo and Ueno. We also compared our one-hour-ahead prediction baselines by evaluating data from 2011.03.01 to 2011.03.07. We can see from Fig.3.4 that our proposed prediction can accurately predict the peaks of rush hours and switch between the

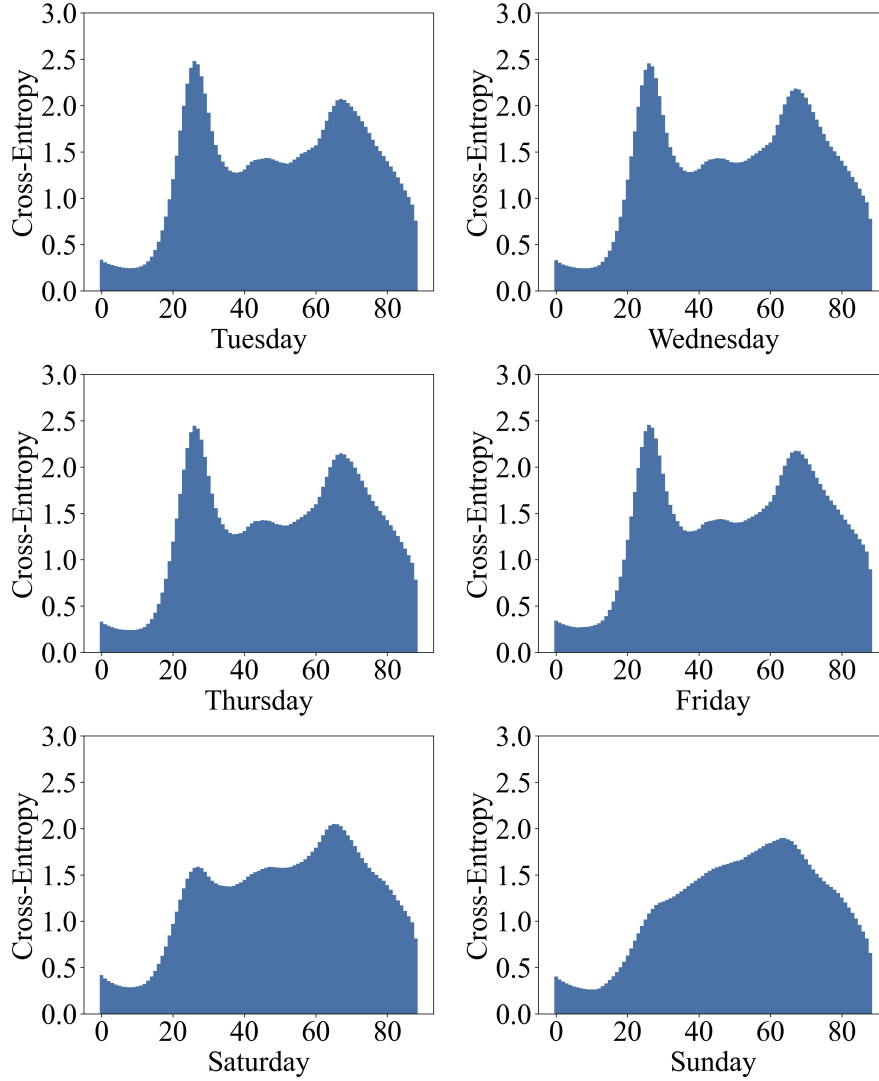


Fig.3.2: Daily Cross-Entropy losses of weekdays and weekends (x-axis refers to the time slot of the day)

weekday and weekend state via prior knowledge automatically, which are difficult for other methods. A more comprehensive quantitative evaluation is given in Table 3.2. From Fig.3.4 and Table.3.2 we found that our proposed model can adapt to mobility patterns not only on weekdays but also on weekends, while baselines *GRU* and *GRU\_crowd\_context* also have similar capability. However, thanks to the complexity of the meta-learning indicator/prior knowledge generator  $\Phi$  in these three models, our proposed method had the best performance for its combination of spatial information and the collective state of all the users. In Table 3.2, our proposed model performed well for aggregation of all train lines and Yamanote Line, though for the Joban line, our model seemed less competitive for unstable fluctuation on

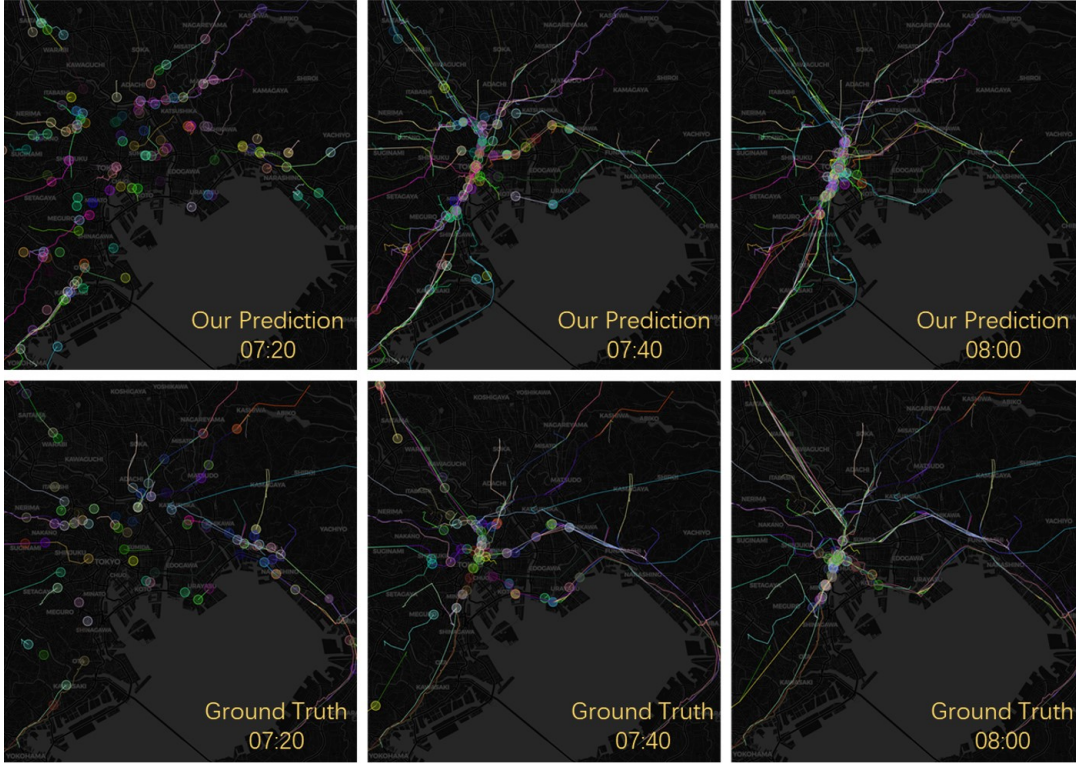


Fig.3.3: Gathering pattern of Tokyo station at morning rush hour

visitors to train stations. We believe the situation happened because the line is relatively independent compared with the Yamamote Line, which means fewer interchange stations make the predictive tasks simpler, and the MAML framework performed better with such simpler tasks.

On the contrary, Fig.3.4 suggests that the baseline *Deepmove*, which shows strong potential in cluster-level prediction of the Section 3.3, failed to generate reliable results during weekends for morning rush hours. Besides, the famous meta-learning structure *MAML* also had a similar problem, and that might due to the setting of the training tasks unsuitable for our spatiotemporal predictive experiments, which resulted in its more minor improvement compared with *GRU* and *GRU\_crowd\_context* baselines.

### 3.5 Evaluation on time efficiency

Before our real-time implementation, we measured the time efficiency performance of the different models in Section 3.4 with a historical trajectory retrieval-based database. Our algorithm was run on a single machine without any acceleration strategies except GPU-accelerated online inference. From Table 3.3, we can see that our proposed model spent



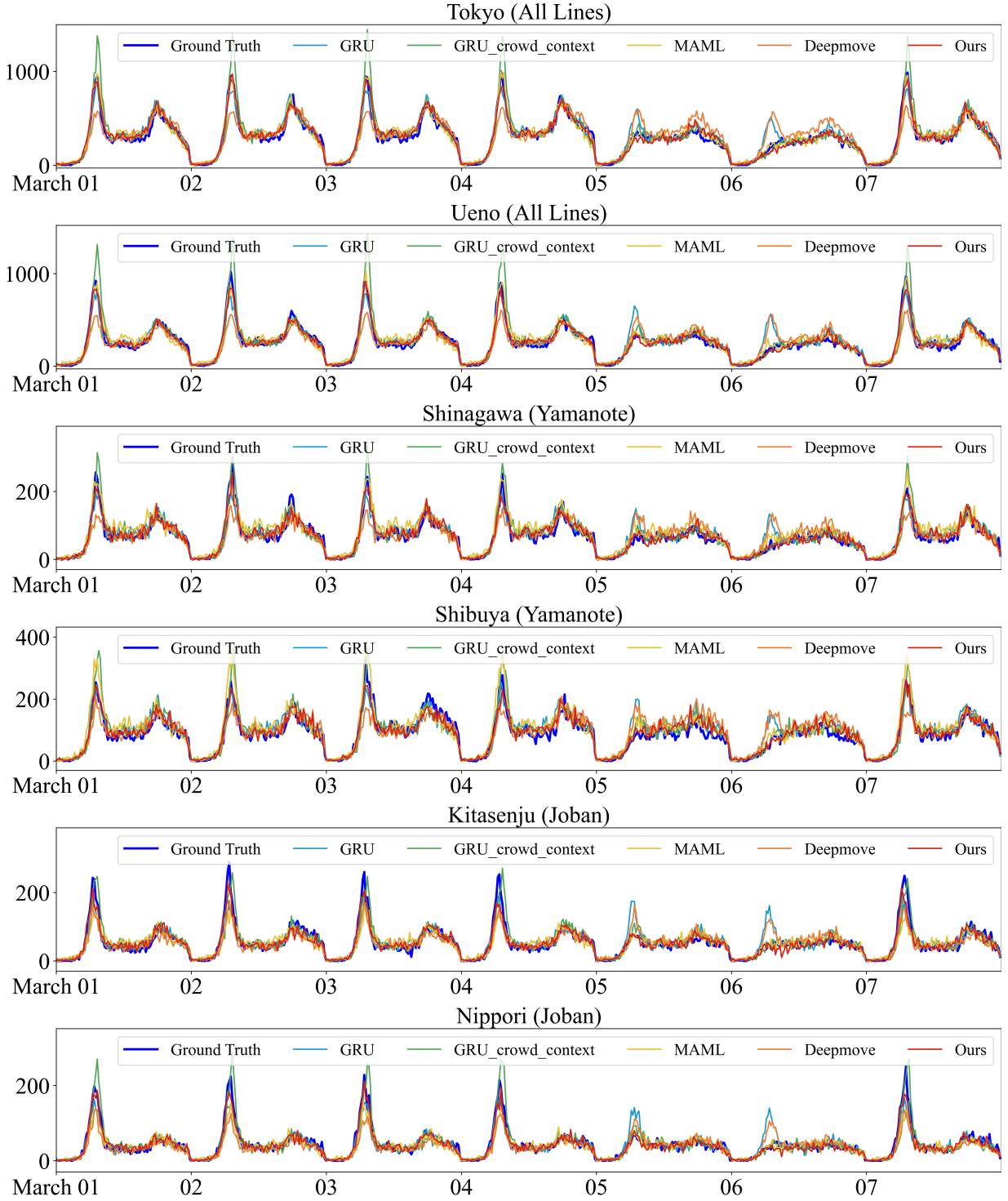


Fig.3.4: Node-level prediction results evaluation

Table3.2: Evaluation on Node-level Prediction

	RMSE / MAE / MAPE				
	GRU	GRU_crowd_context	MAML	Deepmove	Ours
Shinjuku(All lines)	129.1 / 90.4 / 0.4338	143.6 / 92.3 / 0.4009	142.1 / 110.5 / 0.7461	163.1 / 115.0 / 0.5369	<b>81.2 / 63.3 / 0.3654</b>
Tokyo(All lines)	65.7 / 45.6 / 0.7373	88.2 / 51.3 / 0.7917	54.4 / 41.3 / 1.2345	102.4 / 68.7 / 1.1203	<b>47.1 / 35.0 / 0.7081</b>
Shibuya(All lines)	88.3 / 63.2 / 0.3995	91.8 / 62.5 / <b>0.3921</b>	80.2 / 63.0 / 0.6050	108.4 / 76.2 / 0.5258	<b>58.0 / 44.4 / 0.3967</b>
Ueno(All lines)	65.9 / 42.3 / 0.5774	88.3 / 46.5 / 0.5699	58.9 / 47.3 / 0.9877	92.7 / 56.3 / 0.6189	<b>38.8 / 28.9 / 0.5100</b>
Akihabara(All lines)	65.3 / 41.3 / 0.6929	81.2 / 44.5 / 0.6869	49.8 / 38.7 / 1.1636	94.5 / 57.4 / 0.8797	<b>38.7 / 28.6 / 0.6534</b>
Ikebukuro(All lines)	102.0 / 70.4 / 0.5874	111.1 / 73.2 / 0.6028	109.5 / 85.3 / 1.0334	118.2 / 81.9 / 0.7129	<b>67.5 / 52.6 / 0.5690</b>
Shinagawa(All lines)	67.5 / 43.8 / 0.5193	66.7 / 42.6 / 0.4855	67.3 / 54.4 / 1.1068	95.0 / 58.0 / 0.5718	<b>48.1 / 33.0 / 0.4692</b>
Shinjuku(Yamanote)	87.4 / 68.6 / 1.5072	99.3 / 69.5 / 1.5474	104.7 / 74.6 / 1.6982	85.6 / <b>64.6</b> / 1.4354	<b>81.5 / 66.6 / 1.4006</b>
Tokyo(Yamanote)	37.7 / 25.7 / 0.6471	51.5 / 29.8 / 0.7550	42.6 / 27.2 / 0.6575	38.9 / 24.8 / 0.6285	<b>30.1 / 22.1 / 0.5411</b>
Shibuya(Yamanote)	65.0 / 49.5 / 1.1261	74.5 / 50.2 / 1.1502	80.3 / 55.2 / 1.2892	61.4 / <b>45.3</b> / 1.0213	<b>57.5 / 46.1 / 0.9948</b>
Ueno(Yamanote)	35.3 / 23.7 / 0.5996	45.4 / 25.9 / 0.6520	41.1 / 26.4 / 0.6432	34.6 / 22.1 / 0.5612	<b>26.9 / 19.3 / 0.4701</b>
Akihabara(Yamanote)	34.1 / 23.3 / 0.5920	46.8 / 26.9 / 0.6817	38.5 / 24.1 / 0.5932	35.6 / 22.6 / 0.5764	<b>26.4 / 18.6 / 0.4702</b>
Shinagawa(Yamanote)	45.4 / 32.1 / 0.7769	56.6 / 33.7 / 0.8306	56.0 / 39.3 / 0.9080	45.6 / 30.1 / 0.7329	<b>36.3 / 27.2 / 0.6454</b>
Ueno(Joban)	45.1 / 38.7 / 0.7579	49.0 / 41.3 / 0.8343	44.5 / 37.9 / 0.7441	45.8 / 39.3 / 0.7760	<b>43.9 / 36.9 / 0.7095</b>
Nippori(Joban)	39.5 / 29.1 / 0.6951	51.6 / 32.9 / 0.7829	<b>34.8 / 25.6 / 0.5679</b>	41.0 / 28.9 / 0.6780	<b>34.8 / 27.5 / 0.6099</b>
Kitasenju(Joban)	38.9 / 24.2 / 0.6535	46.1 / 26.5 / 0.6847	<b>30.4 / 20.0 / 0.4920</b>	39.6 / 24.1 / 0.6338	33.7 / 22.4 / 0.5640
Matsudo(Joban)	34.5 / 23.8 / 0.6053	36.8 / 24.6 / 0.6031	<b>31.9 / 21.2 / 0.5110</b>	36.2 / 23.8 / 0.6049	35.0 / 25.2 / 0.6119
Kashiwa(Joban)	31.4 / 25.3 / 0.5557	<b>30.7 / 24.9 / 0.5346</b>	31.1 / <b>23.8 / 0.5175</b>	31.9 / 25.9 / 0.5620	30.8 / 25.5 / 0.5300

Table3.3: Average time efficiency of different models

GRU	GRU_crowd_context	MAML	Deepmove	Ours
169.45s	173.88s	160.62s	633.82s	172.04s

about 3 minutes on average to complete one round of prediction with around 250k users, while its GRU-based competitors have a similar performance. The meta-learning structure *MAML* is the fastest model, while the *Deepmove* had the worst time efficiency among all the models. Considering the performance of node-level prediction, our final products, it is clear that *MAML* did not outperform on the accuracy, and *Deepmove* consumed too much time to finish predictive tasks, which is the fatal drawback for implementation to a real-time system.

### 3.6 Evaluation on real-time implementation

In this section, we implemented and evaluated the designed real-time prediction system with our proposed methods with different events to check the flexibility of our framework. We selected four different kinds of events in our datasets: 1) normal weekdays (2011.03.04 Friday), 2) national holidays on weekdays (2011.02.11 Friday), 3) natural disasters (2011.03.11 Friday, Tohoku earthquake), and 4) the next of the disaster (2011.03.12 Saturday). As mentioned in section 2.2.2, the designed system will dynamically modify the model if an irregular loss (cross entropy) is detected. We set the threshold as 0.15, referencing the model's historical performance in Fig.2.9. The data was divided by second and continuously fed to the system as a simulation of a real-time data stream. The results are shown in Fig.3.5.

In Fig.3.5, we found that the designed system can work smoothly to handle the simulated data stream. We tested the update mechanism mentioned in Section 2.2.1.3, and the results shows that the designed real-time adaptation mechanism worked well with our simulated real-time data stream. We evaluated three irregular events, and the system did detect the abnormal predictive model performance. As illustrated in Fig.3.5, the predicted results fluctuated after the update, while the regular day on the left-top of the figure shows similar performance as the offline experiments. We believe these outputs were subjected to the limited information in the historical trajectory database. For instance, after the great Tohoku earthquake, many people failed to take trains as the count number to the Tokyo station was close to 0, while actually our database do not have this kind of trajectories in the afternoon.

Nevertheless, when the predicted results output some movement to retrieve historical trajectories, the mechanism cannot be carried out well because the retrieved trajectories ap-

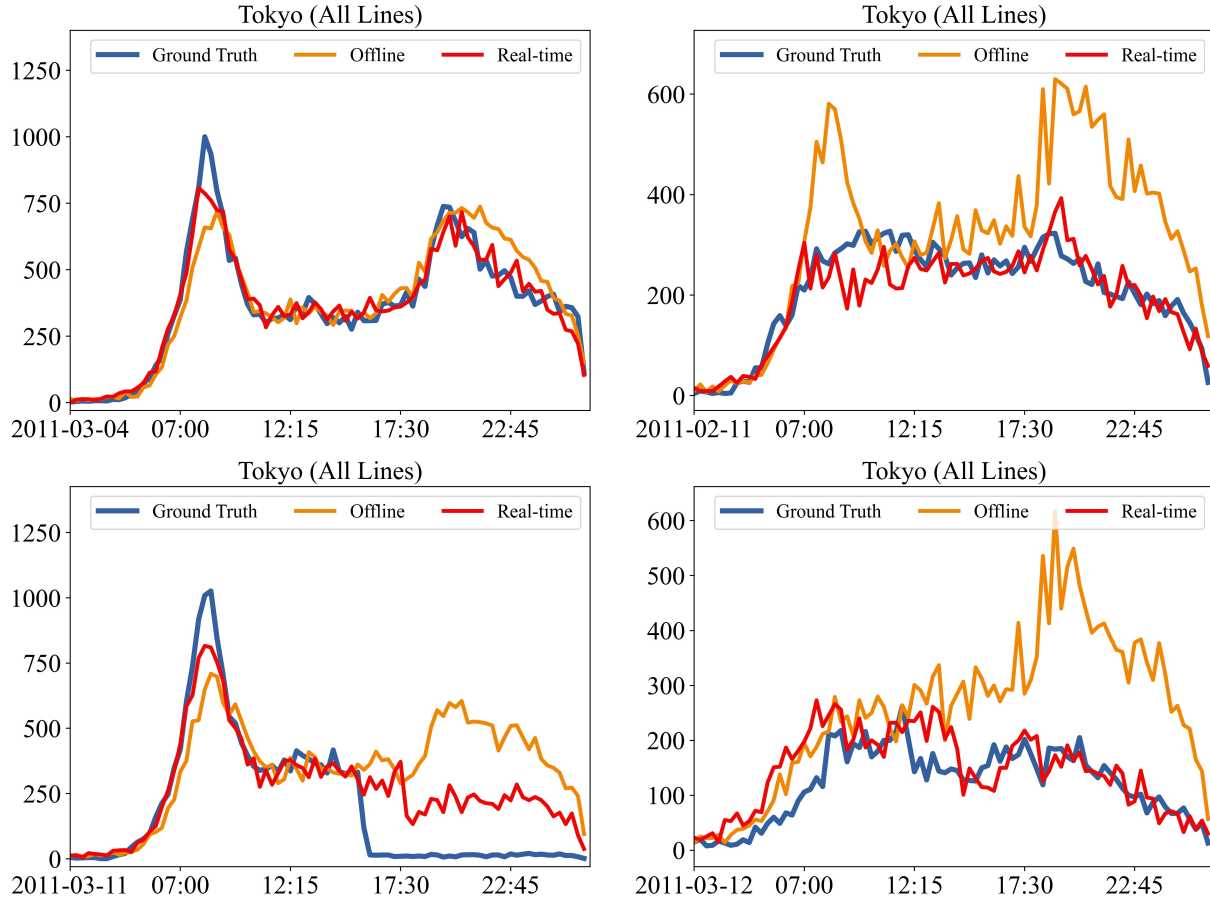


Fig.3.5: Performance of designed real-time system

pear to have an overall tendency to move when the disaster happened in the afternoon. The retrieval-based historical trajectory database lacks human mobility information for disasters, like staying in one place for a long time. Therefore, when new patterns appear in our data stream, the model can be modified to adapt to them, but the database is hard to generate trajectories with new mobility patterns.

## Section 4

# Conclusions and Prospects

### 4.1 Conclusions

This paper proposed a real-time trajectory prediction system structure with the meta-learning paradigm. The meta-learning shows the importance of prior knowledge in multiple predictive tasks, which is spatiotemporal human mobility prediction in this research. The prior knowledge combined with crowd context of aggregated individual hidden states and Flashback spatial attention. Besides, we merged multiple hidden patterns to get a more flexible and efficient prediction procedure. Next, we built a retrieval-based historical trajectory database to query trajectories after getting cluster-level predicted outcomes in our previous step. Based on all mentioned above, we also designed a real-time system and implemented it with python to test if our method can deal with a simulated real-time raw GPS data stream.

In Section 3, we evaluated our proposed model with some intermediate products during designing the prior knowledge generator and some state-of-arts frameworks. We found that crowd context and spatial information fusion will enhance the model's performance with a slight temporal cost, compared with Deepmove, which shows good performance in cluster-level prediction in Table.3.1. However, its low time efficiency was demonstrated in Table.3.3 makes it improper to implement our system setting described in Section 3.2. Moreover, we enriched our predictive results with a retrieval-based historical trajectory database to query similar map-matched trajectories with users' last observed locations. For the trajectories allocated to the transportation networks, we also counted the visit volume in our results to important transportation hubs like train stations in the Tokyo metropolis and compared them with ground truth. As results stressed in Fig.3.4 and Table.3.2, it is obvious that our proposed model outperformed the rest of baselines. We have proved that the introduction of the prior

knowledge generator mentioned in Equation 2.12 benefits the proposed model with the ability to discriminate predictive tasks between predictions under different spatial and temporal conditions, particularly as reflected in the difference between weekends and weekdays. However, in Section 3.6, after we implemented our designed real-time prediction system described in Section 2.2.2, we found that those patterns extracted from our training dataset can be well applied to predictive tasks of normal days. Though the model update mechanism of Section 2.2.1.3 can detect irregularities from the real-time data stream, due to the limitation of the historical trajectory database, the output of such environment did not perform well, and that should be the optimized in our future work.

We would like to note some limitations of the current work: 1) our proposed spatial attention crowd context can describe the global scale mobility patterns for all users so that we can find irregularities such as holidays, and disasters, which are influential enough to change the collective mobility pattern. On the contrary, the local irregularities, and traffic congestion, for instance, will be less significant to be aware of at such a metropolitan scale; 2) the database can not be changed or updated during the prediction procedure, which makes it inflexible to the event. As Fig.3.5 shows, there is still room for improvement in our system performance because of the lack of trajectories generated during irregular events like disasters. 3) python programs the current implementation with a *multiprocessing* package as a temporary solution, which could be optimized with other programming languages like JAVA or C++.

## 4.2 Prospects

The basic idea in our research is the meta-learning paradigm, and we need to evaluate and set a threshold to make sure irregularity happens. In future work, we want to design an indicator to detect global and local irregularity which the latest collected information, such as one-hour data  $TR_{t:t+\Delta T}$  (we used the latest two-hour data in our experiments). Moreover, we want to introduce some new frameworks such as MAML in our baseline, though it performs better in image classification and needs to be carefully designed for spatiotemporal prediction.

Furthermore, the lack of proper historical trajectory will also be considered to solve in the future. Combining real-time traffic information to block some retrieval choices seems to be a good solution. Also, assembling mobility patterns in different events with an ensemble learning paradigm is promising in this study, though we need to produce dummy data to simulate human mobility when different events happen.

In the end, to make our designed system can be more robust to the real-world raw data stream consisting of more users, we are considering introducing mature industrial engines,

---

like *pyspark*[49] or *kafka*[50], to collect and preprocess the received data from the real-time data stream. These tools will ensure that the whole prediction procedure is complete with high time efficiency. In that case, our predictive model development will benefit from a robust system design by other programming languages.

# Acknowledgement

First, I would like to express my appreciation to Prof. Ryosuke Shibasaki for his continuous support and intelligent guidance in my research. He is an excellent director in his field and brings his precious experience and insightful suggestions to make people live in a smarter society. I believe it will be my honor to follow him during the two short years at the University of Tokyo. I still remember the day I received the offer as a research student from Prof. Shibasaki when I was an undergraduate. After that, I passed the entrance exam and became a master's student, which is one of the most wondrous things these years. Besides, I also want to show my gratitude to Associate Professor Xuan Song, who helped me enter the domain of computer science and trained me to become proficient in the tools and skills necessary for future research. He also provided many constructive comments in each of our discussions.

Thanks also to advisors in Shibasaki Lab, especially Zipei Fan and Renhe Jiang, for their kind help and directions during my research. Researcher Fan, whom I came to know about two years ago, did share his reasonable ideas and insightful suggestions not only in the study but also in my life. I will not be able to finish this research without him, who is my teacher and friend.

I will give the same thanks to other laboratory members: Mingxin Zhang, Zhiheng Chen, Chuang Yang, Chuyao Feng, Liqing Xu, Lifeng Lin, Yinghao Liu, Hang Yin, Zekun Cai, Jinyu Chen, Yuhao Yao, Zhiwen Zhang, Peiran Li, Zhaonan Wang, Zhiling Guo, Xiaoya Song, Tianqi Xia, Guangming Wu, Wei Yuan, Qianwei Chen. They are my dear partners and friends during my study in Japan. Also, I believe there are still many future opportunities to share ideas and cooperate with them.

Thanks to my parents and Miss Wenting Zhang, who always accompanied, encouraged, and supported me, no matter what happened. I understand that my study at the University of Tokyo is not easy, but they still show selfless care and help me in many ways.

For these three years, our world profoundly suffered from the world spreading COVID-19. The pandemic changed many people's lives; some failed to make it. I still remember the



days when Wuhan was locked down when I did not believe that the epidemic would change everyone's life after that. I hope my power will help people recover from this pandemic, and our world can be better without wars, conflicts, hunger, and illness.

# Reference

- [1] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, “Characterising the digital twin: A systematic literature review,” *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, 2020.
- [2] A. Lee, K.-W. Lee, K.-H. Kim, and S.-W. Shin, “A geospatial platform to manage large-scale individual mobility for an urban digital twin platform,” *Remote Sensing*, vol. 14, no. 3, p. 723, 2022.
- [3] J. Yuan, Y. Zheng, and X. Xie, “Discovering regions of different functions in a city using human mobility and pois,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 186–194.
- [4] U. Schlink, K. Strebel, M. Loos, R. Tuchscherer, M. Richter, T. Lange, J. Wernicke, and A. Ragas, “Evaluation of human mobility models, for exposure to air pollutants,” *Science of the total environment*, vol. 408, no. 18, pp. 3918–3930, 2010.
- [5] R. Jiang, Z. Wang, Z. Cai, C. Yang, Z. Fan, T. Xia, G. Matsubara, H. Mizuseki, X. Song, and R. Shibasaki, “Countrywide origin-destination matrix prediction and its application for covid-19,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2021, pp. 319–334.
- [6] Z. Fan, C. Yang, Z. Zhang, X. Song, Y. Liu, R. Jiang, Q. Chen, and R. Shibasaki, “Human mobility-based individual-level epidemic simulation platform,” *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 8, no. 3, pp. 1–16, 2022.
- [7] X. Song, Q. Zhang, Y. Sekimoto, and R. Shibasaki, “Prediction of human emergency behavior and their mobility following large-scale disaster,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 5–14.
- [8] M. Luca, G. Barlacchi, B. Lepri, and L. Pappalardo, “Deep learning for human mobility: a survey on data and models,” *arXiv preprint arXiv:2012.02825*, 2020.
- [9] M. S. Nasiri and S. Shokouhyar, “Actual consumers’ response to purchase refurbished

- smartphones: Exploring perceived value from product reviews in online retailing,” *Journal of Retailing and Consumer Services*, vol. 62, p. 102652, 2021.
- [10] P. I. Tebe, G. Wen, J. Li, Y. Yang, W. Tian, J. Chong, and W. Zhang, “5g-enabled medical data transmission in mobile hospital systems,” *IEEE Internet of Things Journal*, 2022.
- [11] Y. Pang, K. Tsubouchi, T. Yabe, and Y. Sekimoto, “Intercity simulation of human mobility at rare events via reinforcement learning,” in *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, 2020, pp. 293–302.
- [12] K. Zhao, S. Tarkoma, S. Liu, and H. Vo, “Urban human mobility data mining: An overview,” in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1911–1920.
- [13] M. Luca, G. Barlacchi, B. Lepri, and L. Pappalardo, “A survey on deep learning for human mobility,” *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–44, 2021.
- [14] 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*, 2021.
- [15] 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.
- [16] G. Dai, X. Hu, Y. Ge, Z. Ning, and Y. Liu, “Attention based simplified deep residual network for citywide crowd flows prediction,” *Frontiers of Computer Science*, vol. 15, no. 2, pp. 1–12, 2021.
- [17] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, “Deepmove: Predicting human mobility with attentional recurrent networks,” in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1459–1468.
- [18] Q. Liu, S. Wu, L. Wang, and T. Tan, “Predicting the next location: A recurrent model with spatial and temporal contexts,” in *Thirtieth AAAI conference on artificial intelligence*, 2016.
- [19] D. Yang, B. Fankhauser, P. Rosso, and P. Cudre-Mauroux, “Location prediction over sparse user mobility traces using rnns: Flashback in hidden states!” in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020, pp. 2184–2190.
- [20] 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.

- [21] G. Kreml, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, *et al.*, “Open challenges for data stream mining research,” *ACM SIGKDD explorations newsletter*, vol. 16, no. 1, pp. 1–10, 2014.
- [22] J. Wang, X. Kong, F. Xia, and L. Sun, “Urban human mobility: Data-driven modeling and prediction,” *Acm Sigkdd Explorations Newsletter*, vol. 21, no. 1, pp. 1–19, 2019.
- [23] M. U. Kraemer, C.-H. Yang, B. Gutierrez, C.-H. Wu, B. Klein, D. M. Pigott, O. C.-. D. W. Group <sup>†</sup>, L. Du Plessis, N. R. Faria, R. Li, *et al.*, “The effect of human mobility and control measures on the covid-19 epidemic in china,” *Science*, vol. 368, no. 6490, pp. 493–497, 2020.
- [24] X. Kong, M. Li, J. Li, K. Tian, X. Hu, and F. Xia, “Copfun: An urban co-occurrence pattern mining scheme based on regional function discovery,” *World Wide Web*, vol. 22, no. 3, pp. 1029–1054, 2019.
- [25] X. Kong, F. Xia, J. Wang, A. Rahim, and S. K. Das, “Time-location-relationship combined service recommendation based on taxi trajectory data,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1202–1212, 2017.
- [26] J. Wang, X. Kong, A. Rahim, F. Xia, A. Tolba, and Z. Al-Makhadmeh, “Is2fun: Identification of subway station functions using massive urban data,” *IEEE Access*, vol. 5, pp. 27 103–27 113, 2017.
- [27] S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo, “Socio-spatial properties of on-line location-based social networks,” in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 5, no. 1, 2011, pp. 329–336.
- [28] C. Song, T. Koren, P. Wang, and A.-L. Barabási, “Modelling the scaling properties of human mobility,” *Nature physics*, vol. 6, no. 10, pp. 818–823, 2010.
- [29] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, “Understanding individual human mobility patterns,” *nature*, vol. 453, no. 7196, pp. 779–782, 2008.
- [30] Y. Qiao, Z. Si, Y. Zhang, F. B. Abdesslem, X. Zhang, and J. Yang, “A hybrid markov-based model for human mobility prediction,” *Neurocomputing*, vol. 278, pp. 99–109, 2018.
- [31] Q. Gao, F. Zhou, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, “Predicting human mobility via variational attention,” in *The World Wide Web Conference*, 2019, pp. 2750–2756.
- [32] Z. Haifeng, Y. Yajie, and Z. Ningbo, “Human mobility prediction based on dbscan and rnn,” in *2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET)*. IEEE, 2021, pp. 146–152.
- [33] M. Katranji, G. Sanmarty, L. Moalic, S. Kraiem, A. Caminada, and F. H. Selem, “Rnn

- encoder-decoder for the inference of regular human mobility patterns,” in *2018 international joint conference on neural networks (IJCNN)*. IEEE, 2018, pp. 1–9.
- [34] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [35] X. Song, H. Kanasugi, and R. Shibasaki, “Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level,” in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016, pp. 2618–2624.
- [36] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.
- [37] D. Kong and F. Wu, “Hst-lstm: A hierarchical spatial-temporal long-short term memory network for location prediction.” in *IJCAI*, vol. 18, no. 7, 2018, pp. 2341–2347.
- [38] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva, “Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips,” *IEEE transactions on visualization and computer graphics*, vol. 19, no. 12, pp. 2149–2158, 2013.
- [39] J. A. Deri, F. Franchetti, and J. M. Moura, “Big data computation of taxi movement in new york city,” in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 2616–2625.
- [40] D. Nallaperuma, R. Nawaratne, T. Bandaragoda, A. Adikari, S. Nguyen, T. Kempitiya, D. De Silva, D. Alahakoon, and D. Pothuhera, “Online incremental machine learning platform for big data-driven smart traffic management,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4679–4690, 2019.
- [41] Z. Fan, X. Song, T. Xia, R. Jiang, R. Shibasaki, and R. Sakuramachi, “Online deep ensemble learning for predicting citywide human mobility,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–21, 2018.
- [42] J. X. Wang, “Meta-learning in natural and artificial intelligence,” *Current Opinion in Behavioral Sciences*, vol. 38, pp. 90–95, 2021.
- [43] V. Fortuin and G. Rätsch, “Deep mean functions for meta-learning in gaussian processes,” *arXiv preprint arXiv:1901.08098*, 2019.
- [44] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.
- [45] A. Nichol and J. Schulman, “Reptile: a scalable metalearning algorithm,” *arXiv preprint*

- arXiv:1803.02999*, vol. 2, no. 3, p. 4, 2018.
- [46] H. Yao, Y. Liu, Y. Wei, X. Tang, and Z. Li, “Learning from multiple cities: A meta-learning approach for spatial-temporal prediction,” in *The World Wide Web Conference*, 2019, pp. 2181–2191.
- [47] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.
- [48] S. Ghemawat and J. Dean, “LevelDB,” <https://github.com/google/leveldb/>, 2011.
- [49] R. K. Mishra and S. R. Raman, “Introduction to pyspark sql,” in *PySpark SQL Recipes*. Springer, 2019, pp. 1–22.
- [50] N. Narkhede, G. Shapira, and T. Palino, *Kafka: the definitive guide: real-time data and stream processing at scale*. ” O’Reilly Media, Inc.”, 2017.