AdaMove: Efficient Test-Time Adaptation for Human Mobility Prediction

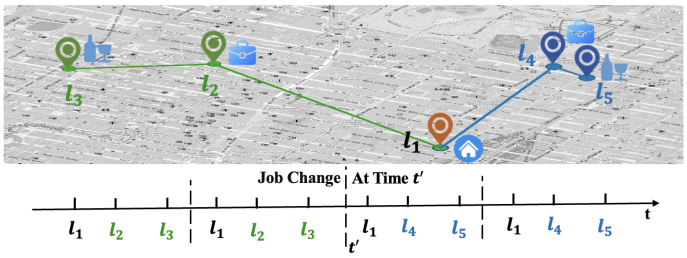
Huaxu Han[1], Shuliang Wang[1], Sijie Ruan[1*], Qianyu Yang[1], Yuxuan Liang[2], Ziqiang Yuan[1], Cheng Long[3], Hanning Yuan[1], Yu Zheng[4,5]
[1]*Beijing Institute of Technology, Beijing, China* [2]*HKUST(GZ), Guangzhou, China*
[3]*Nanyang Technological University, Singapore*
[4]*JD iCity, JD Technology, Beijing, China* [5]*JD Intelligent Cities Research, China*
{hhx7,slwang2011,sjruan,yangqy,ziqiangy,yhn6}@bit.edu.cn;
yuxuanliang@hkust-gz.edu.cn; c.long@ntu.edu.sg; msyuzheng@outlook.com

***Abstract*—Human mobility prediction is a fundamental technique for many urban applications, e.g., location-based recommendation, traffic scheduling, and travel demand prediction. Over the past decades, many methods, e.g., Markov Model, RNN, Transformer, have been leveraged to tackle the problem. However, existing approaches mainly train a supervised model based on an offline training dataset, which overlooks the phenomenon that the mobility behaviors of humans vary across time, and the trained models may not achieve ideal performance when applied to the testing data. To tackle this challenge, in this paper, we propose AdaMove, an efficient Test-Time Adaptive (TTA) model for human mobility prediction. AdaMove has a Preference-aware Test-Time Adaptation module called PTTA, which can adjust the parameters of a trained model based on the input test trajectory such that the model can generalize to the test distribution. In addition, to address the issue of reduced inference efficiency caused by parameter adjustment during the testing phase, AdaMove is equipped with a Lightweight human Mobility prediction model called LightMob, which only requires the recent trajectory as input to accelerate the inference. It is enhanced by historical trajectory knowledge via contrastive learning during the training time, so it has competitive performance compared with existing models. Extensive experiments on three real-world human mobility datasets demonstrate that AdaMove outperforms the best baseline by 9.3% on average in accuracy, and accelerates the inference speed by 28.5% on average compared with the original TTA-based inference.**

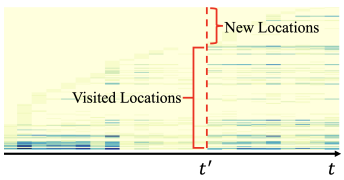
***Index Terms*—human mobility prediction, test-time adaptation, spatio-temporal data mining**

I. Introduction

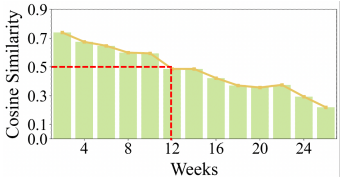
Human mobility prediction is widely studied in trajectory data mining [1]–[3], which predicts the next location based on past trajectories. It serves as an essential building block for many urban applications. For example, residents can benefit from accurate Point of Interest (POI) recommendations based on their historical trajectories [4], the government leverages it to design better transportation strategies to handle crowd aggregations [2], mobile advertisers rely on it to prepare targeted advertisement in advance [5], and ride-sharing platforms can estimate the travel demands based on the prediction results [6].



(a) Temporal Shifted Mobility Pattern due to Job Change.



(b) Mobility Pattern Example.



(c) Mobility Pattern Similarity.

Fig. 1: Temporal Shifts in Human Mobility Data.

Over the past decades, tremendous methods have been developed for human mobility prediction, including Markov model-based methods [7], [8], RNN-based methods [2], [9]–[12], Transformer-based methods [13]–[20], and recent LLM-based methods [21], [22]. Given the strong periodicity patterns in human trajectories, explicitly feeding historical trajectories together with recent trajectories into a model of human mobility prediction can further boost the performance as has been found by many existing studies [2], [5], [11], [21].

While existing studies have been continuously making progress in improving the accuracy of the next location prediction task, they are all developed based on the assumption that the training set and testing set follow the same distribution, which indicates the models trained based on the training set would be directly used to make inference on the testing set. However, this assumption does not always hold as verified by our explorations on real-world human mobility data. In fact, the mobility behaviors of humans often vary across time, as a result, a model that has been trained based on the data (before some time point) would not work effectively on the data afterwards.

The first two authors contributed equally to this work.
*Sijie Ruan is the corresponding author.

For example, as shown in Figure 1(a), Alice frequently followed the mobility pattern $l_1(home) \rightarrow l_2(office) \rightarrow l_3(bar)$ before time $t^{'}$, which is highly regular, making it easy to predict her next location. However, at certain time $t^{'}$, her activities became significantly different due to a change of her job. Her new workplace introduced some new visiting places, altering her mobility pattern to $l_1(home) \rightarrow l_4(office) \rightarrow l_5(bar)$. To demonstrate the phenomenon, we randomly pick a user from a real-world human mobility dataset, i.e., Foursquare[1]. We visualize her visiting frequency of different locations over the course of one year using a heatmap as shown in Figure 1(b), where each pixel represents the number of visits to a certain location in every two weeks. From the heatmap, we can observe that some locations have not been visited in the early period, which implies the location distribution shifts across time. To justify the universality of mobility pattern shifts, we further analyze the mobility pattern similarity across time over all users in the dataset. More specifically, we first calculate the location visit distribution for each user in the earliest three months, and then average those distributions to obtain the *historical mobility distribution*. After that, for every two weeks afterwards, we calculate the *biweekly mobility distribution* in the same way based on data during those two weeks. Finally, we calculate the cosine similarity between those two distributions to quantify the mobility similarity with respect to historical data. We visualize the change of mobility similarity over time in Figure 1(c). As can be observed, with the increase of time, mobility similarity decreases, and by the 12th week, the similarity falls below 50%. Therefore, if we train a model to predict his next location based on the historical trajectories, the model will likely fail to predict accurately due to the lack of the latest information.

Fortunately, since the periodicity pattern in human mobility still exists even in short-term trajectories, it is possible to adjust the parameters of the prediction model based on recent trajectory data to achieve higher prediction accuracy. To this end, in this paper, we propose AdaMove, an efficient test-time Adaptive model for human mobility prediction. AdaMove is composed of a Preference-aware Test-Time Adaptation module, called PTTA, and a Lightweight Mobility prediction model, called LightMob. We explain the rationale of these two modules as follows. PTTA adjusts the parameters of a trained next location prediction model based on input test trajectory, allowing the trained model can accommodate the distribution shifts in the test data and improve the prediction performance. Nevertheless, adjusting model parameters would inevitably increase the time of inference. To further mitigate this issue, LightMob is developed. LightMob is enhanced by the historical trajectory knowledge during training via contrastive learning, which only requires a short recent trajectory as input. Such design makes LightMob highly efficient during inference while maintaining competitive performance compared with existing methods.

Our contributions can be summarized as follows:

- We identify the human mobility shifting issue across time, which is overlooked by existing methods, and present a framework AdaMove to tackle it.
- We propose a preference-aware test-time adaptation module, i.e., PTTA, which leverages the nature of auto-regressive and stable short-term preferences in human trajectories to perform the test time adaptation.
- We propose a lightweight human mobility prediction model, i.e., LightMob, which accelerates the inference speed while retaining competitive prediction accuracy compared to existing models.
- Extensive experiments on three real-world human mobility datasets demonstrate the effectiveness and efficiency of AdaMove. AdaMove outperforms the best baseline by 9.3% on average in terms of accuracy, and accelerates the inference speed by 28.5% on average compared to the original TTA-based inference. We have released the code for public use[2].

II. Preliminaries

In this section, we first give some used definitions, and then formulate the human mobility prediction problem, and finally introduce the basic concepts of test-time adaptation.

A. Definitions

Definition 1 (Spatio-temporal Point). *The spatio-temporal point p_i is a tuple of a timestamp t and a location identifier l, i.e., $p_i = (l, t)$.*

Definition 2 (Trajectory). *The trajectory of a particular person u is a sequence of spatio-temporal points organized chronologically, i.e., $tr^u =< p_1, p_2, \cdots, p_{N_u} >$.*

Definition 3 (Recent Trajectory). *The recent trajectory tr^u_{rec} is a suffix point sequence of tr^u, i.e., $tr^u_{rec} =< p_j, p_{j+1}, \cdots, p_{N_u} >$, which satisfies $t_{N_u} - cT \leq t_j \leq t_{N_u}$, T is the time interval of a session, and c is the number of sessions, i.e., context length.*

Problem Definition. Given historical trajectories of all users $\mathcal{T} = \{tr^1, tr^2, tr^U\}$, learn a human mobility prediction model, which receives a trajectory tr^u of user u, and predicts his/her next visit location $\hat{l}$.

The semantics of locations, such as POI category, may also be beneficial for mobility prediction; however, we did not consider them to maintain consistency with existing work [2], [11], [14], [18]. Note that such information can be easily fused as introduced later.

[1]https://foursquare.com/

[2]https://github.com/hhx7/AdaMove

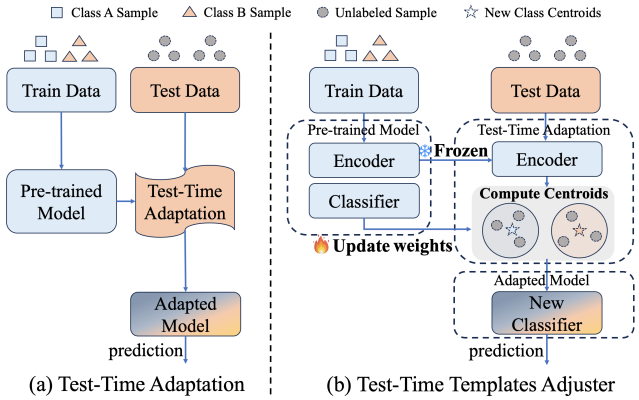


Fig. 2: The Idea of Test-time Adaptation.

B. Test-time Adaptation (TTA)

We first introduce the basic concept of TTA here. As shown in Figure 2(a), the main idea of TTA is to adapt a pre-trained model to make predictions by utilizing the latest knowledge of the test data during the test phase. TTA can be categorized into two main categories: *test-time domain adaptation*, which adapts a pre-trained model on source domain to a target domain at test time; and *test-time distribution adaptation*, which adapts a pre-trained model to a test distribution of the same domain. A comprehensive survey of TTA can be found in [23].

Since we aim to tackle the distribution shifts of predicted next locations, test-time distribution adaptation-based methods are appropriate to be used. Based on whether or not the backward propagation is involved, existing TTA methods can generally be categorized into Gradient-based and Gradient-free methods [24]. Gradient-based methods can be time-consuming at the test time as they need to update parameters of the model. On the other hand, gradient-free methods are more efficient. A representative algorithm of gradient-free methods is Test-Time Templates Adjuster, i.e., T3A [25]. It provides a lightweight yet effective mechanism for handling distribution shifts without completely retraining. The main process of T3A is shown in Figure 2(b). T3A abstracts the classification model into an encoder, which is frozen during the test phase and a classifier, which would be adjusted during the test phase. To adapt to the test data, T3A first constructs a test-time knowledge base (implemented by a dictionary), whose key is one of the class and value is a list initialized with the parameter vector corresponding to that class in the classifier. Then, for each test sample, the model predicts its class and T3A calculates its "importance" to determine whether to keep it. If the sample is important, T3A adds its hidden representation, i.e., the output of the encoder, to the list of predicted class in the knowledge base. After iterating over all test samples, T3A treats the centroid of vectors for each class in the knowledge base as the adjusted parameters of the classifier to make the prediction. The sample importance is characterized by the entropy of the output probability distribution. Intuitively, if the entropy is smaller, prediction is more reliable, thus the sample is more important.

III. Methodology

In this section, we elaborate our proposed Efficient Test-time Adaptive Human Mobility Prediction method, i.e., AdaMove. We first introduce the framework of AdaMove, and then present its two components in details.

A. Framework

AdaMove is composed of a preference-aware test-time adaptation module (PTTA), and a lightweight human mobility prediction model (LightMob). The framework of AdaMove is shown in Figure 3. PTTA makes the human mobility prediction model adapt to the test input trajectory to improve the prediction accuracy. However, such an approach sacrifices the inference efficiency. Therefore, we further propose LightMob, an efficient lightweight human mobility prediction model, to accelerate the inference speed while preserving the performance. In the following, we first explain PTTA, and then introduce LightMob.

B. Preference-aware Test-time Adaptation

Given a well-trained mobility prediction model, Preference-aware Test-time Adaptation module adjusts its parameters based on the input of test sample to make it adapt to the user's short-term preference change.

Since human mobility prediction essentially is a classification problem, a straightforward solution is to adopt an existing test-time classifier adjustment module, e.g., T3A [25] as mentioned in Section II-B. However, existing methods are mainly designed for computer vision tasks, we identify two limitations of those methods: *1) Unreliable Sample Assignment.* As described, the test sample is allocated to the list of the *predicted* class in the knowledge base. However, if the test distribution changes greatly, the predicted class may not be the actual class of the test sample. Such assignment may not benefit or even degrade the original classifier in the later process. *2) Aggressive Sample Filtering.* Whether a sample is added to the list is determined by the entropy of the output distribution. Again, when the OOD issue is severe, even for a sample with correct prediction, the entropy might be large. Though such sample is invaluable for test-time adaptation, it would be filtered in entropy-based filtering strategy. Given aforementioned mechanisms, existing methods are only applicable to handle slight data distribution shifts and would face significant challenges if the shifts are large.

Main Idea. Though aforementioned limitations are difficult to be handled in ordinary data formats, e.g., images, fortunately, we have opportunities to tackle those issues in trajectory data. *Firstly,* trajectories are auto-regressive, i.e., the arbitrary prefix of a trajectory is also a trajectory. A prefix of a trajectory together with its next item can form a labeled sample, which can be used to tackle the first limitation. *Secondly,* the mobility pattern of human in a short time is relatively stable. We can use the similarity of mobility pattern to quantify the importance

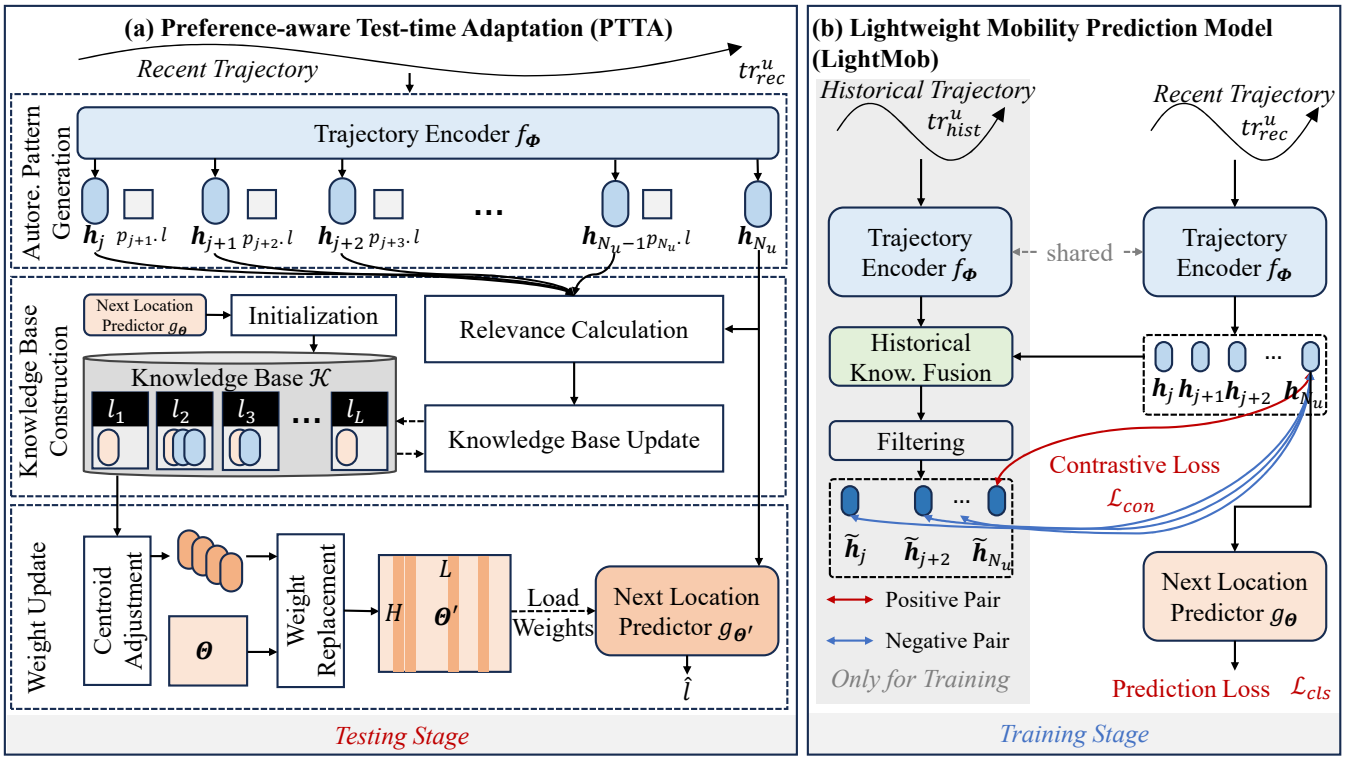


Fig. 3: Framework of AdaMove.

of each sample to tackle the second limitation. To this end, we propose Preference-aware Test-time Adaptation module as shown in Figure 3(a), which is a test-time classifier adjustment module dedicated for trajectories. It consists of three steps, i.e., Labeled Pattern Generation, which generates labeled patterns based on test trajectory input; Knowledge Base Construction, which allocates samples into the dictionary based on the label as well as the preference similarity; and Weight Update, which updates parameters of the output layer for prediction.

Note that, a user's mobility pattern may change over time. Therefore, though the full trajectory of a user is available for us during the test time, we only take his/her recent trajectory tr^u_{rec} for adaptation. As stated in Definition 3, we only require a sequence of spatio-temporal points within the past cT hours to construct tr^u_{rec}, which can be achieved by a sliding window strategy in the memory for real-time applications.

The pseudo code of preference-aware test-time adaptation is given in Algorithm 1. We detail them as follows.

Autoregressive Pattern Generation. Autoregressive Pattern Generation fully exploits the autoregressive nature of the trajectory data, which takes a recent trajectory tr^u_{rec}, and generates labeled patterns based on all trajectory prefixes as well as the next location. For example, if a recent trajectory is $< p_j, p_{j+1}, p_{j+2}, p_{j+3} >$, all its prefixes are $\{< p_j >, < p_j, p_{j+1} >, < p_j, p_{j+1}, p_{j+2} >\}$. We feed them into the trajectory encoder f_Θ of the trained model to obtain the hidden representations, i.e., mobility patterns, i.e., $\{\mathbf{h}_j, \mathbf{h}_{j+1}, \mathbf{h}_{j+2}\}$. We use the mobility patterns as well as the next location to construct the labeled pattern set $\mathcal{P} = \{(\mathbf{h}_j, p_{j+1}.l), (\mathbf{h}_{j+1}, p_{j+2}.l), (\mathbf{h}_{j+2}, p_{j+3}.l)\}$ as shown in Line 1-5.

Knowledge Base Construction. Knowledge Base Construction takes the generated labeled patterns $\mathcal{P}$, and constructs a knowledge base $\mathcal{K}$ with important patterns from labeled patterns for adaptation.

Firstly, we retrieve parameters $\mathbf{\Theta} \in \mathbb{R}^{H \times L}$ of output classification layer g_Θ in the trained model, where H is the dimension of the hidden representation of a trajectory, and L is the number of possible locations for prediction. And then, we use it to initialize the knowledge base $\mathcal{K}$. More specifically, $\mathcal{K}$ is implemented by a dictionary, where each key is a possible location l, and the value is a list initialized with corresponding column in $\mathbf{\Theta}$, i.e., $\boldsymbol{\theta}_l \in \mathbb{R}^H$, as shown in Line 6-8.

After that, we assign labeled patterns $\mathcal{P}$ into the knowledge base $\mathcal{K}$ based on the label of the pattern. As previously mentioned, the mobility pattern of human in a short time is relatively stable. Therefore, we use the similarity between the mobility pattern of the test sample and its sub-trajectory to filter out irrelevant patterns. More specifically, we use $\mathbf{h}_{N_u}$ to denote the mobility pattern of test sample, and $\mathcal{P}_l$ to denote patterns whose label, i.e., next location, is l. For each possible location l in $\mathcal{K}$, we only keep top-M most similar patterns in addition to $\boldsymbol{\theta}_l$, i.e., $\mathcal{K}_l = \mathcal{P}_l^M \cup \{\boldsymbol{\theta}_l\}$ (Line 9-16), where $|\mathcal{P}_l^M| = \texttt{min}(M, |\mathcal{P}_l|)$ and $\forall \mathbf{h} \in \mathcal{P}_l^M$, $\forall \mathbf{h}' \in \mathcal{P}_l - \mathcal{P}_l^M$, $\texttt{sim}(\mathbf{h}_{N_u}, \mathbf{h}) > \texttt{sim}(\mathbf{h}_{N_u}, \mathbf{h}')$, where $\texttt{sim}(\cdot, \cdot)$ is the cosine similarity, i.e.,

$$\texttt{sim}(\mathbf{h}_i, \mathbf{h}_j) = \frac{\mathbf{h}_i^\top \mathbf{h}_j}{\|\mathbf{h}_i\|\|\mathbf{h}_j\|} \tag{1}$$

Weight Update. Weight Update takes the knowledge base $\mathcal{K}$, and gives the adjusted parameter weights $\mathbf{\Theta}'$ of the classification layer. Since the knowledge base contains the recent mobility patterns for each possible next location, the idea is to use the mean vector to update the corresponding weights, which represents the common mobility pattern for each possible next location as shown in Line 17-21. More specifically, the vector $\boldsymbol{\theta}_l$ in the l-th column of $\mathbf{\Theta}$ is replaced by $\boldsymbol{\theta}_l^{'}$:

$$\boldsymbol{\theta}_l^{'} = \frac{1}{|\mathcal{K}_l|} \sum_{\mathbf{h} \in \mathcal{K}_l} \mathbf{h} \tag{2}$$

Note that, for locations do not involve new patterns, i.e., $\mathcal{K}_l$ only contains its original weights, we do not need to update weights for that column.

Inference. After the weight updating, we can make the next location prediction for tr^u based on the adjusted model. We freeze the parameters $\mathbf{\Phi}$ in the trajectory encoder $f_{\mathbf{\Phi}}$, which serves as a feature extractor to obtain the hidden representation $\mathbf{h}_{N_u}$, and use the updated predictor $g_{\mathbf{\Theta}}$ to make the next location prediction, i.e.,

$$\hat{l} = \arg\max g_{\mathbf{\Theta}'}(\mathbf{h}_{N_u}) \tag{3}$$

Complexity Analysis. We give a complexity analysis of the Preference-aware Test-time Adaptation.

In the first step, we iterate over all prefixes of a recent trajectory to generate the labeled patterns. Therefore, the time complexity is $\mathcal{O}(N_u)$. In the second step, we iterate over all patterns, which also takes $\mathcal{O}(N_u)$, and each pattern might be added to the knowledge base if its similarity is within the top-M patterns of the same next location. The top-M list can be implemented by a priority queue, in which case, the queue updating only takes $\mathcal{O}(logM)$. Therefore, the time complexity of the second step is $\mathcal{O}(N_u logM)$. In the third step, we iterate over $\mathcal{O}(L)$ locations to update parameters. The updated parameters can be calculated in $\mathcal{O}(M)$. Therefore, the time complexity of the third step is $\mathcal{O}(LM)$.

Since M and L are all constants, the overall complexity of Preference-aware Test-time Adaptation is $\mathcal{O}(N_u)$, which is practical.

C. Lightweight Mobility Prediction Model

The mobility behavior of human usually has periodicity. For example, a person tends to visit the same location at the same day of week and time of day. Therefore, existing studies, e.g., DeepMove [2], LSTPM [11], and LLMMob [21], find it beneficial to consider historical trajectories when predicting the human mobility. However, since historical trajectories can be arbitrarily long, taking it as the input of the model introduces great computational costs. The situation becomes more severe when we have to perform the adaptation at the test time, which makes the inference process inefficient.

Main Idea. To mitigate this issue, a lightweight model, which can also take historical trajectories into consideration, is desirable. To this end, we propose Lightweight Mobility Prediction Model (LightMob). The main idea of LightMob is to use a simple sequential base model to make the prediction during the test time, which only takes a recent trajectory as input, while the knowledge from historical trajectories is incorporated during the training time. By removing the branch of encoding the historical trajectory at the test time, the computational costs can be reduced. The basic idea is that if a base model have *implicitly* incorporated the historical knowledge, its representation should be similar to the representation

Algorithm 1: Preference-aware Test-time Adaptation.

Input : A test trajectory tr^u, a well-trained human mobility prediction model, which is composed of a trajectory encoder $f_{\mathbf{\Phi}}$ and a predictor $g_{\mathbf{\Theta}}$.
Output: The predicted next location $\hat{l}$.

/* Step 1. Autoregressive Pattern Generation. */
$tr_{rec}^u \leftarrow$ get recent trajectory from tr^u;
$\mathcal{D} \leftarrow \emptyset$;
for $k \in 1, \cdots, |tr_{rec}^u| - 1$ **do**
  $\mathbf{h}_k \leftarrow f_{\mathbf{\Phi}}(tr_{rec}^u[1:k])$;
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{h}_k, p_{k+1}.l)\}$;
/* Step 2. Knowledge Base Construction. */
$\mathcal{K} \leftarrow \emptyset$;
for $\boldsymbol{\theta}_l \in \mathbf{\Theta}$ **do**
  $\mathcal{K}_l \leftarrow \{\boldsymbol{\theta}_l\}$;
$\mathbf{h}_{N_u} \leftarrow f_{\mathbf{\Phi}}(tr_{rec}^u[1:|tr_{rec}^u|])$;
for $(\mathbf{h}_j, l_{j+1}) \in \mathcal{D}$ **do**
  if $|\mathcal{K}_{l_{j+1}}| < M$ **then**
    $\mathcal{K}_{l_{j+1}} \leftarrow \mathcal{K}_{l_{j+1}} \cup \{\mathbf{h}_j\}$;
  else
    $s^{min}, \mathbf{h}^{min} \leftarrow$ find the pattern with the minimum similarity in $\mathcal{K}_{l_{j+1}}$;
    if $sim(\mathbf{h}_{N_u}, \mathbf{h}_j) > s^{min}$ **then**
      Replace $\mathbf{h}^{min}$ with $\mathbf{h}_j$;
/* Step 3. Weight Updating. */
$\mathbf{\Theta}' \leftarrow \mathbf{\Theta}$;
for $\mathcal{K}_l \in \mathcal{K}$ **do**
  if $|\mathcal{K}_l| > 1$ **then**
    $\boldsymbol{\theta}_l^{'} \leftarrow \frac{1}{|\mathcal{K}_l|} \sum_{\mathbf{h} \in \mathcal{K}_l} \mathbf{h}$;
    $\mathbf{\Theta}'[:, l] \leftarrow \boldsymbol{\theta}_l^{'}$;
$\hat{l} \leftarrow \arg\max g_{\mathbf{\Theta}'}(\mathbf{h}_{N_u})$;
return $\hat{l}$

given by a model that *explicitly* incorporates the historical knowledge. We find such learning objective is possible to be achieved via contrastive learning [26]. For example, CLIP [27], a well-known multi-modality model trained via contrastive learning, also has two encoders. The training objective of CLIP is to pull close representations from both encoders for positive pairs and push away representations from both encoders for negative pairs, which ultimately enhances the performance of individual encoders. Such observation motivates us to design a constrastive historical knowledge incorporation method as shown in Figure 3. In the following, we first introduce the base model, then illustrate how to construct the contrastive samples based on historical and recent trajectories, and finally explain how the model is trained.

Base Model. The base model takes the recent trajectory tr_k^u, and predicts the next visit location $\hat{l}$. We decompose the base model into two parts, i.e., *Trajectory Encoder* $f_{\mathbf{\Phi}}$, which takes tr_k^u as input, and gives a hidden representation $\mathbf{h}_{N_u}$, and *Next Location Predictor* $g_{\mathbf{\Theta}}$, which predicts the next visit location $\hat{l}$. We detail them as follows.

- **Trajectory Encoder $f_{\mathbf{\Phi}}$.** In trajectory encoder, we first embed each spatio-temporal point p_i into dense representation, i.e., $\mathbf{e}_i$. Following existing work [2], for each point, we encode its location l_i, time t_i and user u. Both location and user are represented by ID, which can be directly fed into separate embedding layers. As for time t_i, considering different mobility patterns in workdays and on weekends, we encode t_i into 48 discrete slots. We use $[0, 23]$ to encode hour of day in workdays, and $[24, 47]$ to encode hour of day on weekends. Finally, we use the concatenation of those information to represent the embedding of each spatio-temporal point p_i:

$$\mathbf{e}_i = [\texttt{Emb}(l_i); \texttt{Emb}(Code(t_i)); \texttt{Emb}(u)] \tag{4}$$

  where ; denotes the concatenation operation, $Code(\cdot)$ is the coding scheme for time. The semantic information can also be concatenated here if available.

  Then, to capture the sequential dependency among trajectory points, we can use arbitrary sequential modeling layer, e.g., GRU [28], LSTM [29], Transformer [30], to encode the embedding sequence of trajectory points.

$$\mathbf{h}_{N_u} = \texttt{SeqEncoder}(\{\mathbf{e}_j, \mathbf{e}_{j+1}, \cdots, \mathbf{e}_{N_u}\}) \tag{5}$$

- **Next Location Predictor $g_{\mathbf{\Theta}}$.** The next location predictor transforms the hidden representation $\mathbf{h}_{N_u}$ into the output space via a fully connected (FC) layer parameterized by $\mathbf{\Theta}$ followed by a softmax layer, i.e.,

$$\hat{\mathbf{Y}} = \texttt{Softmax}(\texttt{FC}(\mathbf{h}_{N_u})) \tag{6}$$

  Then, the location with the maximum predicted probability is regarded as the predicted next visit location, i.e., $\hat{l} = \arg\max \hat{\mathbf{Y}}$.

Constrastive Sample Generation. Constrastive Sample Generation generates positive and negative samples for constrastive learning based on historical trajectories and recent trajectories.

First, we treat all spatio-temporal points before $tr_{rec}^u = \{p_j, p_{j+1}, \cdots, p_{N_u}\}$ as the historical trajectory of u, i.e., $tr_{hist}^u = \{p_1, p_2, \cdots, p_{j-1}\}$. Then, we feed tr_{hist}^u and tr_{rec}^u to the shared trajectory encoder $f_{\mathbf{\Phi}}$ to obtain the hidden representations for all points, i.e., $\mathbf{H}_{hist}^u = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{j-1}\}$ and $\mathbf{H}_{rec}^u = \{\mathbf{h}_j, \mathbf{h}_{j+1}, \cdots, \mathbf{h}_{N_u}\}$.

Since we want the representations of models with and without historical knowledge similar, we now need to obtain representations for points in tr_{rec}^u that incorporate the historical trajectory knowledge. Inspired by [2], we use the attention mechanism to fuse such knowledge. More specifically, we apply linear transformations to obtain key matrix $\mathbf{K} \in \mathbb{R}^{(j-1)\times d_k}$ and value matrix $\mathbf{V} \in \mathbb{R}^{(j-1)\times d_v}$ based on $\mathbf{H}_{hist}^u$, and query matrix $\mathbf{Q} \in \mathbb{R}^{(N_u-j+1)\times d_k}$ based on $\mathbf{H}_{rec}^u$. Then we calculate the attention weights $\mathbf{A} \in \mathbb{R}^{(N_u-j+1)\times(j-1)}$ based on aforementioned three matrices:

$$\mathbf{A} = \texttt{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right), \tag{7}$$

where $\sqrt{d_k}$ is the scaling factor. Finally, we use the weighted sum to obtain the history-enhanced representations $\tilde{\mathbf{H}}_{rec}^u = \{\tilde{\mathbf{h}}_j, \tilde{\mathbf{h}}_{j+1}, \cdots, \tilde{\mathbf{h}}_{N_u}\}$:

$$\tilde{\mathbf{H}}_{rec}^u = \mathbf{A}\mathbf{V} \tag{8}$$

With history-enhanced representations $\tilde{\mathbf{H}}_{rec}^u$, we are now able to construct the positive and negative pairs for learning. We treat $\mathbf{h}_{N_u}$ and its history-enhanced representation $\tilde{\mathbf{h}}_{N_u}$ as positive pair, and regard $\mathbf{h}_{N_u}$ and some other history-enhanced representations $\tilde{\mathbf{O}}_{rec}^u = \{\tilde{\mathbf{h}}_q | \tilde{\mathbf{h}}_q \in \tilde{\mathbf{H}}_{rec}^u - \{\tilde{\mathbf{h}}_{N_u}\} \wedge p_{q+1}.l \neq p_{N_u+1}.l\}$ as negative pairs. Note that, we filter out some prefix trajectory representations whose next location is the prediction target to avoid the confusion of the learning.

Model Training. With constrastive samples generated in the previous step as well as the label of the next location of the trajectory, on one hand, we want LightMob predict the next location as accurate as possible, and on the other hand, we encourage LightMob to produce similar trajectory representations whether the historical trajectories are given or not, which is a typical multi-task training objective.

To optimize the representation similarity objective, we employ InfoNCE loss [26], which is commonly used in constrastive learning. InfoNCE maximizes the similarity between positive pairs and minimizes the similarity between negative pairs:

$$\mathcal{L}_{con} = -\log \frac{\exp(\texttt{sim}(\mathbf{h}_{N_u}, \tilde{\mathbf{h}}_{N_u}))}{\sum_{\tilde{\mathbf{h}} \in \tilde{\mathbf{O}}_{rec}^u} \exp(\texttt{sim}(\mathbf{h}_{N_u}, \tilde{\mathbf{h}}))} \tag{9}$$

To optimize the prediction accuracy objective, we use the classic cross entropy loss:

$$\mathcal{L}_{cls} = -\sum_{l=1}^{L} y_l \log(\hat{y}_l) \tag{10}$$

TABLE I: Data Statistics after Pre-processing.

Dataset	NYC	TKY	LYMOB
City	New York	Tokyo	CityD
Time	2012/04 - 2013/02	2012/04 - 2013/02	75 days
#. of Users	637	1,843	500
#. of Loc.	4,713	7,736	5,906
#. of Traj.	50,720	314,202	467,899

where L is the number of possible locations.

Therefore, the final training objective of LightMob is a hybrid loss function, which can be optimized via stochastic gradient decent [31].

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{con} \tag{11}$$

where λ is a trade-off hyperparameter. λ controls to what extent the historical patterns should be memorized to avoid the overfitting.

IV. Experiments

In this section, we conduct extensive experiments on three real-world human mobility datasets to evaluate the effectiveness and efficiency of AdaMove. Our experiments cover following five parts:

- *Performance Comparison*. We compare AdaMove with eight representative baselines over four evaluation metrics. The experiments results show the superiority of the proposed method.
- *Ablation Study*. We study the effectiveness of different components of AdaMove.
- *Parameter Sensitivity*. We study how different hyperparameter settings affect on our model.
- *Efficiency*. The experiment verifies the efficiency of AdaMove.
- *Case Study*. A case study is further conducted to illustrate the effectiveness of AdaMove to tackle the issue of human mobility behavior change across time.

A. Experimental Setup

Datasets & Data Pre-processing. We evaluate AdaMove as well as baselines on three real-world datasets, i.e., NYC, TKY and LYMOB. NYC and TKY were collected from Foursquare in the city of New York and Tokyo between 2012 and 2013 [32]. These datasets record users' check-in sequences, capturing the locations they visited on the platform. LYMOB [33] was an anonymized human trajectory dataset in grid cell level provided by LY Corporation released in 2024[3]. The dataset contains individuals' trajectories across a 75 day period collected from four undisclosed, highly populated metropolitan area. Due to the cost of massive API calling for LLM-based baselines, we randomly choose trajectories of 500 users from CityD for comparison.

Following existing pre-processing step [2], locations visited by fewer than 10 users would be filtered out to avoid noise. Then, we segment each user's trajectory into distinct sessions, where each session includes the locations visited within a time window of $T = 72$hours. A session contains fewer than 5 locations are also excluded. Furthermore, inactive users, whose trajectory comprises fewer than 5 sessions, are removed. Table I summarizes the statistical details of each dataset after the data pre-processing.

Based on the pre-processed trajectory datasets, for each user, the earliest 70% sessions of each user are used for training, the next 10% sessions are for validation, and the last 20% sessions are for testing. In each train/val/test dataset, we use the sliding window strategy to create the data samples.

Evaluation Metrics. We utilize two types of commonly used evaluation metrics from previous work [16]–[18]: Recall (*Rec@K*) and MRR to assess the performance of all models. Recall measures the proportion of true positive samples among all positive samples. Specifically, *Rec@K* considers only the top-K locations predicted by the model. In our evaluation, we use $K = \{1, 5, 10\}$. Notably, *Rec@1* is equivalent to accuracy. Mean Reciprocal Rank, i.e., MRR, is another widely adopted metric in recommendation systems, used to evaluate the ranking accuracy of model predictions for user queries. A higher *MRR* indicates better ranking quality, enabling users to find their desired results easier. In our evaluation, we specifically compute *MRR@10*, which considers the ranking accuracy within the top 10 predicted locations.

Baselines. We select the following baselines in the field of human mobility prediction:

- **LSTM** [34]: This is the classic variant of the RNN model which has shown strong pattern learning ability in handling sequential data.
- **DeepMove** [2]: The method utilizes the RNN-based method to extract user's preference from historical and current trajectory. And predict the next location by considering user's long-term preferences with attention mechanism.
- **LSTPM** [11]: This method consists of a nonlocal network for long-term preference modeling and a geo-dilated RNN for short-term preference learning.
- **STAN** [14]: This method utilizes a bi-layer attention architecture that first aggregates spatio-temporal correlation within user trajectory and then recalls the target with consideration of personalized item frequency.
- **GETNext** [16]: This method utilizes a global trajectory flow map and a novel Graph Enhanced Transformer model to better leverage extensive collaborative signals.
- **CLSPRec** [35]: This method leverages contrastive learning to learn a shared Transformer-based trajectory encoder to encode both long-term and short-term preferences for next-location prediction.
- **MHSA** [17]: This method leverages a multi-head self-attentional neural network to capture location transition patterns across diverse contexts.
- **MCLP** [18]: This method predicts the next location for individuals, where it explicitly models the user preference and the next arrival time as a a context.

[3]https://zenodo.org/records/14219563

- **LLM-Mob** [21]: We also choose a recent open-source LLM-based model for comparison. LLM-Mob introduces concepts of historical and contextual stays to capture the long- and short-term dependencies in human mobility.

Implementations. All experiments are conducted on Ubuntu 20.04 with an NVIDIA GeForce 2080Ti GPU. We implement AdaMove based on the PyTorch 2.1.0 and Python 3.9. Besides, we use LibCity [36] to implement DeepMove, LSTPM and STAN, and use codes released by authors for other baselines.

Hyperparameters & Training Details. We train our model using the Adam optimizer [31]. The maximum number of training epochs is set to 30, and the batch size is 50. The learning rate is initialized to $1e^{-2}$ and decayed according to average accuracy. The learning rate decay follows a schedule where it reduces proportionally with improvements in accuracy. The training is early stopped when the learning rate reaches $1e^{-4}$. Recall that AdaMove receives c consecutive sessions to form the recent trajectory input. In the training set, $c = 1$ on all datasets, while in validation/testing set, we set c to a larger value, since larger c indicates more data available for adaptation. By default, in validation/testing set, c is set to 5, 6, 5 in NYC, TKY, LYMOB, respectively. In AdaMove, we set the embedding dimension of the location, time and user ID to $\{48, 8, 16\}$, and employ an LSTM as the default trajectory encoder. The capacity M of the knowledge base for each location is set to 5. The trade-off hyperparameter λ in LightMob is set to 0.8, 0.2, 0.6 in NYC, TKY and LYMOB, respectively. For baselines, hyperparameters are set according to the best performance on the validation set.

B. Performance Comparison

Table II presents the performance of various methods for human mobility prediction. From the results in Table II, we have following observations:

- LSTM, GETNext, and MCLP predicts the next location based on a recent trajectory. GETNext outperforms LSTM by utilizing a global trajectory flow map, which captures more spatial information. The improvement brought by MCLP is limited due to the unreliable prediction of next arrival time.
- DeepMove, LSTPM, STAN, MHSA and CLSPRec all take both a historical trajectory and a recent trajectory as input, which demonstrate superior learning capabilities compared to single-branch methods. In particular, MHSA uses a multi-head self-attention neural network to model contextual relationships, outperforming other baselines across three datasets.
- LLM-Mob also follows the two-branch structure, while it is a language-based model. Though it shows some prediction ability, its performance is mediocre on Rec@1, since it is not fine-tuned on the training data. Nevertheless, LLM-Mob demonstrates competitive performance on other metrics.
- Our proposed model outperforms all baselines across all datasets and metrics. It effectively captures long-term patterns from historical trajectories while adapting to recent behavior patterns, even when faced with OOD data. The results demonstrate that our model outperforms the best baseline by 9.3% on average in Rec@1. Note that, the improvement of metrics except for Rec@1 on LYMOB is smaller than other two datasets, which might be attributed to the shorter time span of LYMOB (75 days), i.e., the smaller distribution shifts. But the results also indicate that AdaMove can improve the prediction performance even for small distribution shifts.

TABLE II: Model Performance on Different Datasets

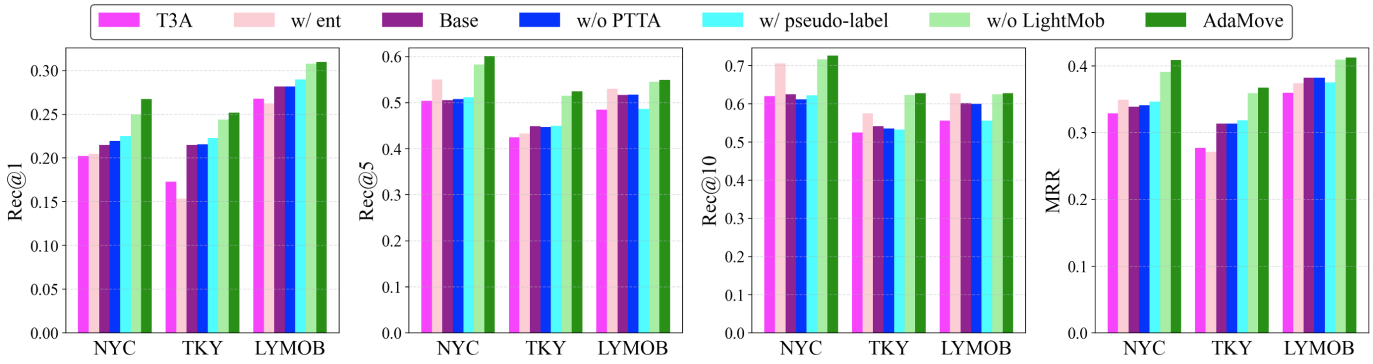
Dataset	Method	Rec@1	Rec@5	Rec@10	MRR
NYC	LSTM	0.2156	0.5022	0.6041	0.3373
	DeepMove	0.2317	0.5324	0.6478	0.3594
	LSTPM	0.2068	0.5212	0.6484	0.3256
	STAN	0.1513	0.4025	0.5242	0.2579
	GETNext	0.1947	0.4976	0.6370	0.3286
	CLSPRec	0.1725	0.3099	0.3579	0.2303
	MCLP	0.1832	0.4818	0.6156	0.3097
	MHSA	0.2250	0.5151	0.6137	0.3528
	LLM-Mob	0.1929	0.5182	0.6430	0.3289
	AdaMove (Ours)	**0.2707**	**0.6044**	**0.7298**	**0.4112**
TKY	LSTM	0.2137	0.4478	0.5377	0.3128
	DeepMove	0.2339	0.4879	0.5826	0.3408
	LSTPM	0.2111	0.4699	0.5700	0.3367
	STAN	0.2031	0.4295	0.5384	0.3153
	GETNext	0.2168	0.4841	0.5809	0.3343
	CLSPRec	0.2201	0.4128	0.4664	0.3005
	MCLP	0.1556	0.4153	0.5224	0.2643
	MHSA	0.2379	0.4899	0.5901	0.3531
	LLM-Mob	0.1626	0.4307	0.5263	0.2733
	AdaMove (Ours)	**0.2518**	**0.5248**	**0.6282**	**0.3672**
LYMOB	LSTM	0.2817	0.5170	0.6023	0.3825
	DeepMove	0.2932	0.5284	0.6148	0.3936
	LSTPM	0.2854	0.5230	0.6123	0.3895
	STAN	0.2785	0.5064	0.5963	0.3724
	GETNext	0.2867	0.5246	0.6094	0.3895
	CLSPRec	0.2912	0.5145	0.5847	0.3904
	MCLP	0.1781	0.4739	0.5710	0.3027
	MHSA	0.2973	0.5455	0.6332	0.4130
	LLM-Mob	0.2131	0.4906	0.5983	0.3272
	AdaMove (Ours)	**0.3125**	**0.5513**	**0.6340**	**0.4149**

C. Ablation Study

To further examine the effectiveness of each component in our model, we perform a series of ablation studies over all datasets and assess performance on four metrics.

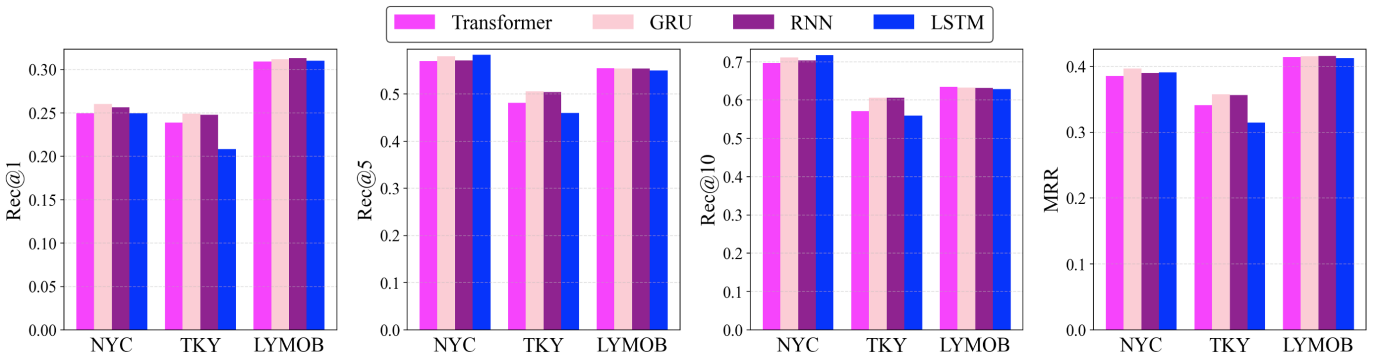
Impact of Different Components. Our model comprises two key components: (1) *lightweight human mobility prediction model (LightMob)* and (2) *preference-aware test-time adaptation (PTTA)*. To assess the contribution of each component to the overall performance, we design following variants: (1) *w/o LightMob*: This variant removes the attention mechanism and contrastive learning loss, namely it only uses base model to perform the adaptation; (2) *w/o PTTA*: This variant excludes the process of updating the classifier weights during the test phase; (3) *Base Model*: this variant directly leverages the base model to make the prediction, which essentially is the LSTM baseline.

Figure 4 illustrates the performance of these variants. *LightMob* is designed to implicitly learn the mobility knowledge from historical trajectories to capture the long-term behaviors



(a) Rec@1 on Different Datasets (b) Rec@5 on Different Datasets (c) Rec@10 on Different Datasets (d) MRR on Different Datasets

Fig. 4: Ablation on Different Model Variants.



(a) Rec@1 on Different Datasets (b) Rec@5 on Different Datasets (c) Rec@10 on Different Datasets (d) MRR on Different Datasets

Fig. 5: Ablation on Different Trajectory Encoders.

of human. *PTTA* focuses on adapting to changes in user preferences by capturing short-term patterns from the most recent trajectories.

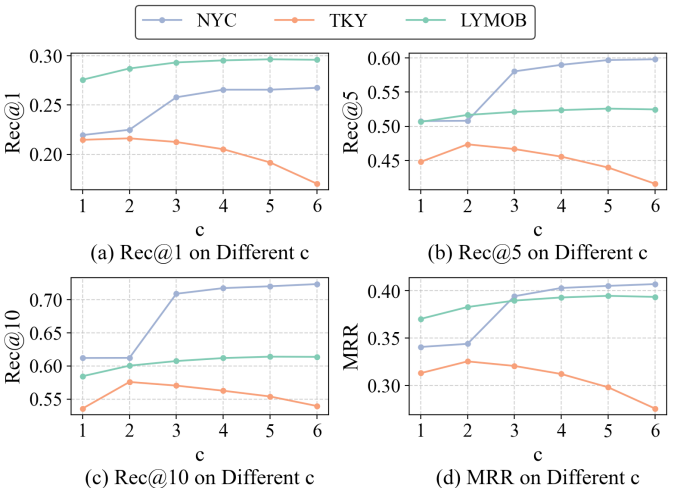
The results indicate that both *w/o LightMob* and *w/o PTTA* outperform *Base Model*, demonstrating the importance of each component in improving performance. However, the performance drop for *w/o PTTA* is more pronounced compared to *w/o LightMob*, highlighting that addressing distribution shifts is a more critical factor for maintaining model accuracy. Finally, the full model, i.e., AdaMove, achieves superior performance compared to *w/o LightMob* across all metrics, emphasizing the effectiveness of *LightMob* to learn historical trajectory knowledge implicitly via the contrastive learning. These findings confirm the complementary roles of *LightMob* and *PTTA* in enhancing the overall performance of the model.

Comparison with T3A. Compared to T3A [25], we propose two key improvements. Firstly, we leverage a new sample importance calculation strategy. Secondly, we use actual labels from test trajectory input instead of pseudo-labels predicted by the model. As can be observed in Fig. 4, AdaMove outperforms T3A by 32.07% on average in terms of Rec@1, which implies the necessity of our dedicated PTTA to handle the distribution shifts in human mobility data. We further analyze those two improvements in detail.

(1) *Impact of Different Sample Importance Calculation Strategy:* Specifically, we implement the *w/ ent* variant, which calculates prediction entropy, instead of using similarity, when storing recent patterns into the knowledge base. The results indicate that the similarity-based strategy consistently outperforms prediction entropy across all metrics. This suggests that similarity is a more effective approach for constructing new pattern sets. In contrast, the performance of the prediction entropy strategy appears to heavily depend on the reliability of the base model. However, the base model proves to be unreliable when encountering out-of-distribution samples, further highlighting the advantages of the similarity-based approach.

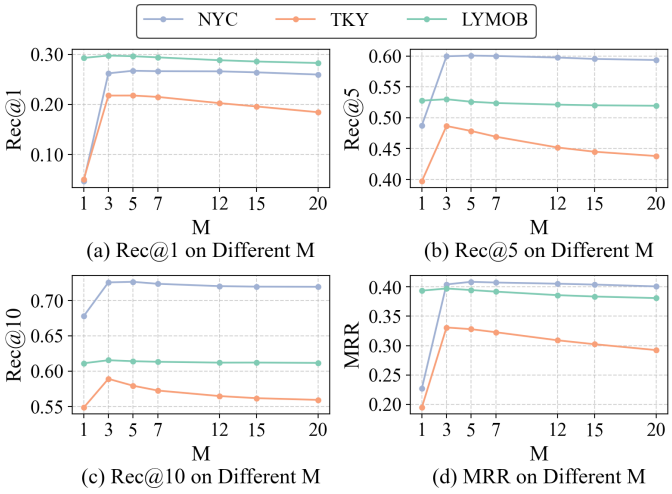
(2) *Impact of Pseudo-Label:* We now further implement the *w/ pseudo-label* variant, where pseudo-labels are used to collect hidden states instead, i.e., the patterns are stored to the knowledge base based on the predicted location rather than the actual next location. Figure 4 shows that the performance of the *w/ pseudo-label* variant drops compared to *AdaMove* after incorporating pseudo-labels. This result highlights the importance of using real labels, as actual locations can calibrate the shifted patterns during the test phase—addressing a major shortcoming of T3A.

Impact of Different Trajectory Encoders. In Figure 5, we analyze the impact of different encoders when extracting



(a) Rec@1 on Different c (b) Rec@5 on Different c (c) Rec@10 on Different c (d) MRR on Different c

Fig. 6: Impact of the Number of Sessions c.



(a) Rec@1 on Different M (b) Rec@5 on Different M (c) Rec@10 on Different M (d) MRR on Different M

Fig. 7: Impact of Capacity of the Knowledge Base M.

hidden states from trajectories. We evaluate four encoders: RNN, LSTM, GRU, and Transformer. RNN-based models are configured with the same hidden size, while Transformer is set as a two-layer architecture with 8 attention heads.

Overall, the performance of our model varies depending on the encoder used. The results indicate that GRU outperforms the other encoders, delivering the best performance. In contrast, Transformer performs worse than RNN-based models across all metrics. Simpler models exhibit higher performance, likely due to the sparsity of trajectory data, which may prevent the Transformer from fully leveraging its advantages.

D. Parameter Sensitivity

We further perform parameter sensitivity analysis on critical hyperparameters, including the number of sessions c used to prepare the short-term trajectory during the testing phase and the maximum number M of stored hidden states for each location y^k in the PTTA module.

Impact of the Number of Sessions c. From Figure 6, we observe that the model's performance initially improves as the number c of sessions increases. However, a larger c contributes little to performance on the NYC and LYMOB dataset and can even result in a decline in performance on the TKY dataset. The difference in performance between the datasets may be due to a more pronounced distribution shift in the TKY dataset. As the number of sessions increases, short-term patterns become less distinct, and the trajectory includes too many diverse patterns for the same location. This causes the hidden states to align more closely with historical patterns rather than recent ones. Overall, capturing short-term patterns becomes challenging when using an excessive number of sessions, leading the model to struggle with adapting to new patterns under distribution shifts.

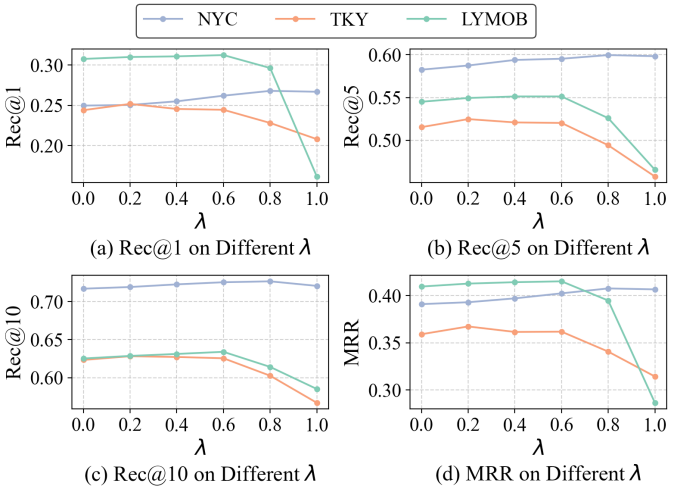
Impact of Capacity of the Knowledge Base M. Figure 7 illustrates the performance of AdaMove when we vary the capacity of the knowledge base for each location M from 1 to 20. As can be observed, the effect of M is prominent on TKY and NYC. Initially, with the increase of M, all metrics become higher. However, when M is larger than 3, the performance drops gradually. Such phenomenon is consistent with our common sense, since if the capacity is too small, the samples can be used for adaptation is quite limited, which cannot well-handle the issue of the distribution shifts. However, a large M is also not ideal, which would make the knowledge base contain many patterns that are not relevant enough, and bring noises for adaptation. We find LYMOB is not sensitive to M because of similar mobility patterns in a short time period.

Impact of λ. The parameter λ controls the influence of the contrastive loss on the total loss in the LightMob module, which determines the extent historical information affects the model. We vary λ from 0 to 1.0 on all datasets, and report the performance in Figure 8. As shown in the figure, three datasets exhibit a consistent trend where the performance improves as λ increases, up to a certain point. Beyond this threshold, the performance starts to decline. But we find the optimal value of λ varies depending on the dataset. Intuitively, a higher λ implies a greater emphasis on historical trajectories. However, if there is a significant shift between the distributions of current and historical trajectories, a lower λ value should be used to avoid overfitting to historical data.

E. Model Efficiency

To evaluate the efficiency of AdaMove, we compare AdaMove with a variant that explicitly feeds historical trajectories into the model during the test time. To implement, we choose the representative baseline, DeepMove [2], which simultaneously receives a current trajectory and a historical trajectory as input. We equip it with our proposed PTTA module, and name the resulted variant as *DeepTTA*.

Before comparing their efficiency, we first report their performance differences. As shown in Figure 9, we observe that our model achieves competitive performance against DeepTTA. Intuitively, DeepTTA should achieve better performance since it incorporates historical trajectories during infer-



(a) Rec@1 on Different λ (b) Rec@5 on Different λ (c) Rec@10 on Different λ (d) MRR on Different λ

Fig. 8: Impact of the Parameter λ.

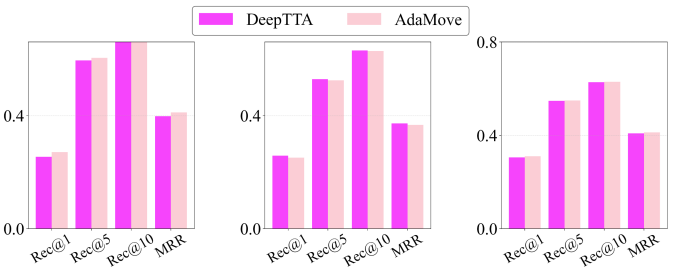
ence, whereas AdaMove does not. However, the experimental results show that the performance of DeepTTA is on par with AdaMove, and notably, AdaMove even slightly outperforms DeepTTA on NYC and LYMOB. This further demonstrates the effectiveness of contrastive learning in memorizing historical trajectory knowledge.

The inference efficiency of both methods is reported in Table III, where the average inference time for an sample is reported. As can be observed, AdaMove outperforms DeepTTA by 30.4%, 10.1%, 45.2%, demonstrating the efficiency of our approach. The efficiency improvement on LYMOB is the greatest, since the trajectories in LYMOB is denser, which takes longer time for DeepTTA to encode the historical trajectories during the test time.

TABLE III: Computational Costs on Different Datasets.

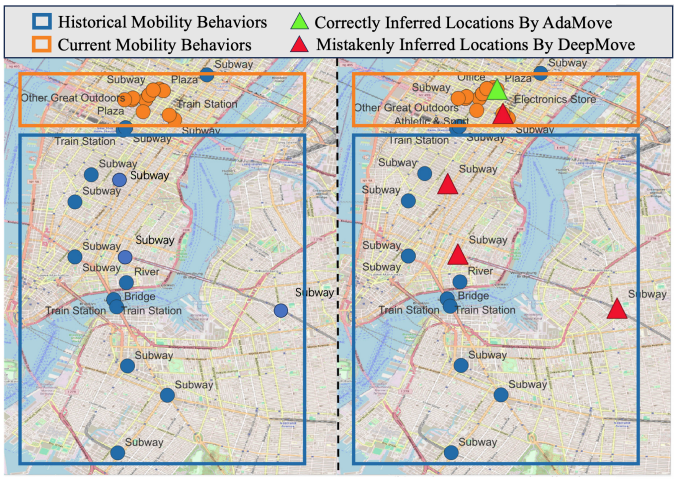
Dataset	Average Time (ms)		Improve
	DeepTTA	AdaMove	
NYC	17.1	11.9	30.4%
TKY	35.5	31.9	10.1%
LYMOB	35.6	19.5	45.2%

Based on aforementioned experiments, we can now reach a conclusion that AdaMove not only achieves a higher accuracy than baselines benefited from PTTA module, but also improves



(a) NYC (b) TKY (c) LYMOB

Fig. 9: AdaMove vs DeepTTA on Different Datasets.



(a) Distribution Shift of Mobility Behaviors (b) Adapt to New Distribution

Fig. 10: A Case Study of a User's Mobility Data in NYC.

the adaptation efficiency given LightMob model.

F. Case Study

We further conduct a case study to illustrate the adaptation ability of AdaMove. We pick a user from NYC dataset, and divided his/her check-in points into two parts, i.e., before and after Jan. 1st, 2013, to represent historical and current mobility behaviors, respectively. We visualize the distributions of his/her check-in points on the map (via sampling) as shown in Figure 10(a). Current check-ins are mainly distributed in the upper region and have more diversity in categories of visited POIs, while historical check-ins are mainly distributed the in lower region. It suggests that mobility patterns vary greatly. We randomly pick four trajectories after Jan. 1st, 2013, whose ground-truth next location is at the green triangle in Figure 10(b). We feed those trajectories into AdaMove and DeepMove to make prediction. The results show that AdaMove correctly predicts the next location, while DeepMove fails to make the prediction. Though we can observe that for one time, the prediction by DeepMove is quite close to the ground-truth, it still fails to predict it correctly due the large distribution shifts.

V. Related Work

In this section, we review literature related to our work, which covers the existing methods for next location prediction as well as handling distribution shifts in trajectory modeling.

A. Next Location Prediction

Next location prediction (a.k.a., human mobility prediction, next POI recommendation) has been extensively studied in the past decade. Depending on whether the target time of the next location is given or not, it is divided into two settings. The former mainly focuses on modeling the sequential behaviors, while the latter is able to incorporate more temporal information.

Though the input of different work might vary slightly, one of the most fundamental problem in next location prediction is to learn the sequential dependency among trajectory points. Early work seeks for statistical methods to predict the next location. For example, GTS-LP [37] mines sequential mobility patterns in historical trajectories, and formulate the next location prediction as a pattern matching problem. PFMC-LR [7] and NLPMM [8] leverage Markov Models to capture sequential dependency. Given the emergence of deep learning, various deep models have been widely adopted to tackle the next location prediction problem. For example, ST-RNN [9], SERM [10], DeepMove [2], LSTPM [11] and FlashBack [12] use Recurrent Neural Networks (RNN), e.g., LSTM, GRU, to encode the trajectory sequence. Transformer and its variants are exploited by GeoSAN [13], STAN [14], MobTCast [15], GETNext [16], MHSA [17], MCLP [18], ROTAN [19], and CLSPRec [35].

Though deep neural networks have the ability to memorize the historical experiences, given the strong periodicity of human trajectories, feeding historical trajectories explicitly into the model is proven to further improve the performance [2]. Among aforementioned methods, [2], [5], [11], [21], [35] all adopt such kind of strategy. DeepMove [2], LSTPM [11], Jeon, et al [5] and CLSPRec [35] employ attention-like methods to fuse historical trajectories. Note that although CLSPRec [35] also leverages contrastive learning to train a mobility prediction model. It differs significantly from us from the aspect of learning purpose. It aims to learn a shared encoder that can be used to encode both historical and recent trajectories, while we aim to memorize the knowledge of historical mobility patterns and make our model lightweight at the test time.

Existing work has made significant progress, which ought to perform well in regular cases. Nevertheless, the parameters of aforementioned models during the test time are frozen, which indicates their performance might degrade when human behaviors change across time. Besides, feeding historical trajectories into the model introduces additional computational overheads, which makes the inference time longer.

In addition to the above expert models, there has been a surge in the development of large foundation models for spatio-temporal data mining [38], such as prompt-based mobility prediction models [21], trajectory foundation models trained on ride-hailing datasets [39], or even billion-scale worldwide trajectory datasets [22], [40]. Due to their extensive capacity, these models excel at inference on previously unseen trajectory data. In contrast, our approach proposes a new test-time adaptation method that enables expert models to generalize to out-of-distribution data, while significantly reducing computational costs.

B. Knowledge Transfer in Spatio-temporal Data

In recent years, due to the pursuit of model generalization, knowledge transfer has increasingly drawn the attention in the community of spatio-temporal data mining. Representative methods of knowledge transfer include transfer learning, domain adaptation, meta-learning and domain generalization [41], [42].

Transfer learning and domain adaptation aim to transfer knowledge learned from data-sufficient domain/distribution to a data-scarce domain/distribution. For example, FLORAL [43] transfers the air quality prediction knowledge from a data sufficient-city to another data-scarce city. MobiTran [44] and COLA [45] transfer human mobility knowledge to a new city. Test-Time Adaptation (TTA) [23] introduced in Section II-B can be considered as a special case to transfer the knowledge, which finishes its adaptation to new domain/distribution at the test time.

Meta-learning is another way to achieve knowledge transfer. For instance, MetaST [1] learns a crowd flow prediction model from multiple source cities that can generalize well to a target city. MetaSTP [46] and MetaSTP$^+$ [47] learns a meta-predictor based on historical delivery events at various locations, which can accurately predict service time based on limited observations.

Domain generalization, i.e., out-of-distribution (OOD) generalization, different from aforementioned methods, cannot access target domain and aims to generalize to unseen distributions. Some work learns invariant representations to generalize to test domain, e.g., disentangled representation learning [48] and causal representation learning [49], [50] for spatio-temporal prediction.

Different from these work, we introduce TTA [23] into the human mobility prediction, which is the first of this kind.

VI. Conclusion

In this paper, we identify issue of the distribution shifts of human mobility data across time, and propose a test-time adaptive human mobility prediction model, i.e., AdaMove, to tackle it. AdaMove is composed of a preference-aware test-time adaptation module, i.e., PTTA, which adapts to the trajectory distribution during the test time, and a lightweight mobility prediction model, i.e., LightMob, to mitigate the efficiency reduction due to the adaptation. Extensive experiments on real-world human mobility datasets demonstrate that AdaMove outperforms the best baseline by 9.3% on average in accuracy, and accelerates the inference speed by 28.5% on average compared with the original TTA-based inference.

In this study, we mainly focus on tackling the distribution shifts in human mobility data and improving the adaptation efficiency without considering interactions between users and semantics of geographical locations. In the future, we aim to extend the base model in AdaMove to a more powerful lightweight model that can distill knowledge comprehensively, e.g., teacher-student model.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62306033, 42371480), and Guizhou Provincial Key Technology R&D Program (No. PA[2025]002).

REFERENCES

[1] H. Yao, Y. Liu, Y. Wei, X. Tang, and Z. Li, "Learning from multiple cities: A meta-learning approach for spatial-temporal prediction," in *WWW*, 2019, pp. 2181–2191.

[2] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "Deepmove: Predicting human mobility with attentional recurrent networks," in *WWW*, 2018, pp. 1459–1468.

[3] S. Ruan, C. Long, J. Bao, C. Li, Z. Yu, R. Li, Y. Liang, T. He, and Y. Zheng, "Learning to generate maps from trajectories," in *AAAI*, vol. 34, no. 01, 2020, pp. 890–897.

[4] J. Long, T. Chen, Q. V. H. Nguyen, and H. Yin, "Decentralized collaborative learning framework for next poi recommendation," *TOIS*, vol. 41, no. 3, pp. 1–25, 2023.

[5] J. Jeon, S. Kang, M. Jo, S. Cho, N. Park, S. Kim, and C. Song, "Lightmove: A lightweight next-poi recommendation fortaxicab rooftop advertising," in *CIKM*, 2021, pp. 3857–3866.

[6] L. Chen, Q. Zhong, X. Xiao, Y. Gao, P. Jin, and C. S. Jensen, "Price-and-time-aware dynamic ridesharing," in *ICDE*, 2018, pp. 1061–1072.

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

[8] M. Chen, Y. Liu, and X. Yu, "Nlpmm: A next location predictor with markov modeling," in *PAKDD*. Springer, 2014, pp. 186–197.

[9] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *AAAI*, vol. 30, no. 1, 2016.

[10] D. Yao, C. Zhang, J. Huang, and J. Bi, "Serm: A recurrent model for next location prediction in semantic trajectories," in *CIKM*, 2017, pp. 2411–2414.

[11] K. Sun, T. Qian, T. Chen, Y. Liang, Q. V. H. Nguyen, and H. Yin, "Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation," in *AAAI*, vol. 34, no. 01, 2020, pp. 214–221.

[12] D. Yang, B. Fankhauser, P. Rosso, and P. Cudre-Mauroux, "Location prediction over sparse user mobility traces using rnns," in *IJCAI*, 2020, pp. 2184–2190.

[13] D. Lian, Y. Wu, Y. Ge, X. Xie, and E. Chen, "Geography-aware sequential location recommendation," in *KDD*, 2020, pp. 2009–2019.

[14] Y. Luo, Q. Liu, and Z. Liu, "Stan: Spatio-temporal attention network for next location recommendation," in *WWW*, 2021, pp. 2177–2185.

[15] H. Xue, F. Salim, Y. Ren, and N. Oliver, "Mobtcast: Leveraging auxiliary trajectory forecasting for human mobility prediction," *NIPS*, vol. 34, pp. 30 380–30 391, 2021.

[16] S. Yang, J. Liu, and K. Zhao, "Getnext: trajectory flow map enhanced transformer for next poi recommendation," in *SIGIR*, 2022, pp. 1144–1153.

[17] Y. Hong, Y. Zhang, K. Schindler, and M. Raubal, "Context-aware multi-head self-attentional neural network model for next location prediction," *Transportation Research Part C: Emerging Technologies*, vol. 156, p. 104315, 2023.

[18] T. Sun, K. Fu, W. Huang, K. Zhao, Y. Gong, and M. Chen, "Going where, by whom, and at what time: Next location prediction considering user preference and temporal regularity," in *KDD*, 2024, pp. 2784–2793.

[19] S. Feng, F. Meng, L. Chen, S. Shang, and Y. S. Ong, "Rotan: A rotation-based temporal attention network for time-specific next poi recommendation," in *KDD*, 2024, pp. 759–770.

[20] Y. Chang, J. Qi, Y. Liang, and E. Tanin, "Contrastive trajectory similarity learning with dual-feature attention," in *ICDE*. IEEE, 2023, pp. 2933–2945.

[21] X. Wang, M. Fang, Z. Zeng, and T. Cheng, "Where would i go next? large language models as human mobility predictors," *arXiv preprint arXiv:2308.15197*, 2023.

[22] L. Gong, Y. Lin, X. Zhang, Y. Lu, X. Han, Y. Liu, S. Guo, Y. Lin, and H. Wan, "Mobility-llm: Learning visiting intentions and travel preferences from human mobility data with large language models," *NIPS*, 2024.

[23] J. Liang, R. He, and T. Tan, "A comprehensive survey on test-time adaptation under distribution shifts," *IJCV*, pp. 1–34, 2024.

[24] S. Niu, C. Miao, G. Chen, P. Wu, and P. Zhao, "Test-time model adaptation with only forward passes," *arXiv preprint arXiv:2404.01650*, 2024.

[25] Y. Iwasawa and Y. Matsuo, "Test-time classifier adjustment module for model-agnostic domain generalization," *NIPS*, vol. 34, pp. 2427–2440, 2021.

[26] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*. PMLR, 2021, pp. 8748–8763.

[28] 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.

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

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu, "Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 129–142, 2014.

[33] T. Yabe, K. Tsubouchi, T. Shimizu, Y. Sekimoto, K. Sezaki, E. Moro, and A. Pentland, "Yjmob100k: City-scale and longitudinal dataset of anonymized human mobility trajectories," *Scientific Data*, vol. 11, no. 1, p. 397, 2024.

[34] A. Graves, "Supervised sequence labelling with recurrent neural networks," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012, pp. 5–13.

[35] C. Duan, W. Fan, W. Zhou, H. Liu, and J. Wen, "Clsprec: Contrastive learning of long and short-term preferences for next poi recommendation," in *CIKM*, 2023, pp. 473–482.

[36] J. Wang, J. Jiang, W. Jiang, C. Li, and W. X. Zhao, "Libcity: An open library for traffic prediction," in *SIGSPATIAL*. New York, NY, USA: Association for Computing Machinery, 2021, p. 145–148.

[37] J. J.-C. Ying, W.-C. Lee, and V. S. Tseng, "Mining geographic-temporal-semantic patterns in trajectories for location prediction," *TIST*, vol. 5, no. 1, pp. 1–33, 2014.

[38] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen, "Foundation models for time series analysis: A tutorial and survey," in *KDD*, 2024, pp. 6555–6565.

[39] Y. Lin, T. Wei, Z. Zhou, H. Wen, J. Hu, S. Guo, Y. Lin, and H. Wan, "Trajfm: A vehicle trajectory foundation model for region and task transferability," *arXiv preprint arXiv:2408.15251*, 2024.

[40] Y. Zhu, J. J. Yu, X. Zhao, X. Wei, and Y. Liang, "Unitraj: Universal human trajectory modeling from billion-scale worldwide traces," *arXiv preprint arXiv:2411.03859*, 2024.

[41] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and S. Y. Philip, "Generalizing to unseen domains: A survey on domain generalization," *TKDE*, vol. 35, no. 8, pp. 8052–8072, 2022.

[42] T. Qian, Y. Chen, G. Cong, Y. Xu, and F. Wang, "Adaptraj: a multi-source domain generalization framework for multi-agent trajectory prediction," in *ICDE*. IEEE, 2024, pp. 5048–5060.

[43] Y. Wei, Y. Zheng, and Q. Yang, "Transfer knowledge between cities," in *KDD*, 2016, pp. 1905–1914.

[44] T. He, J. Bao, R. Li, S. Ruan, Y. Li, L. Song, H. He, and Y. Zheng, "What is the human mobility in a new city: Transfer mobility knowledge across cities," in *WWW*, 2020, pp. 1355–1365.

[45] Y. Wang, T. Zheng, Y. Liang, S. Liu, and M. Song, "Cola: Cross-city mobility transformer for human trajectory simulation," in *WWW*, 2024, pp. 3509–3520.

[46] S. Ruan, C. Long, Z. Ma, J. Bao, T. He, R. Li, Y. Chen, S. Wu, and Y. Zheng, "Service time prediction for delivery tasks via spatial meta-learning," in *KDD*, 2022, pp. 3829–3837.

[47] S. Wang, Q. Yang, S. Ruan, C. Long, Y. Yuan, Q. Li, Z. Yuan, J. Bao, and Y. Zheng, "Spatial meta learning with comprehensive prior knowledge injection for service time prediction," *TKDE*, 2024.

[48] Y. Du, X. Guo, H. Cao, Y. Ye, and L. Zhao, "Disentangled spatiotemporal graph generative models," in *AAAI*, vol. 36, no. 6, 2022, pp. 6541–6549.

[49] Z. Zhou, Q. Huang, K. Yang, K. Wang, X. Wang, Y. Zhang, Y. Liang, and Y. Wang, "Maintaining the status quo: Capturing invariant relations for ood spatiotemporal learning," in *KDD*, 2023, pp. 3603–3614.

[50] Y. Xia, Y. Liang, H. Wen, X. Liu, K. Wang, Z. Zhou, and R. Zimmermann, "Deciphering spatio-temporal graph forecasting: A causal lens and treatment," *NIPS*, vol. 36, 2024.