

# PLMTrajRec: A Scalable and Generalizable Trajectory Recovery Method with Pre-trained Language Models

Tonglong Wei\*  
School of Computer and Information  
Technology, Beijing Jiaotong  
University  
Beijing, China  
{weitonglong}@bjtu.edu.cn

Yan Lin\*  
Youfang Lin  
School of Computer and Information  
Technology, Beijing Jiaotong  
University  
Beijing, China  
{ylincs,yflin}@bjtu.edu.cn

Shengnan Guo†  
School of Computer and Information  
Technology, Beijing Jiaotong  
University  
Beijing, China  
guoshn@bjtu.edu.cn

Jilin Hu  
School of Data Science and  
Engineering, East China Normal  
University  
Shanghai, China  
jlhu@dase.ecnu.edu.cn

Haitao Yuan  
Gao Cong  
Nanyang Technological University  
Singapore  
haitao.yuan,gaocong@ntu.edu.sg

Huaiyu Wan  
School of Computer and Information  
Technology, Beijing Jiaotong  
University  
Beijing, China  
hywan@bjtu.edu.cn

## Abstract

Spatiotemporal trajectory data is crucial for various applications. However, issues such as device malfunctions and network instability often cause sparse trajectories, leading to lost detailed movement information. Recovering the missing points in sparse trajectories to restore the detailed information is thus essential. Despite recent progress, several challenges remain. First, the lack of large-scale dense trajectory data makes it difficult to train a trajectory recovery model from scratch. Second, the varying spatiotemporal correlations in sparse trajectories make it hard to generalize recovery across different sampling intervals. Third, the lack of location information complicates the extraction of road conditions for missing points.

To address these challenges, we propose a novel trajectory recovery model called PLMTrajRec. It leverages the scalability of a pre-trained language model (PLM) and can be fine-tuned with only a limited set of dense trajectories. To handle different sampling intervals in sparse trajectories, we first convert each trajectory’s sampling interval and movement features into natural language representations, allowing the PLM to recognize its interval. We then introduce a trajectory encoder to unify trajectories of varying intervals into a single interval and capture their spatiotemporal relationships. To obtain road conditions for missing points, we propose an area flow-guided implicit trajectory prompt, which models road conditions by collecting traffic flows in each region. We also introduce a road condition passing mechanism that uses observed points’ road conditions to infer those of the missing points. Experiments on two public trajectory datasets with three sampling intervals each demonstrate the effectiveness, scalability, and generalization ability of PLMTrajRec.

## Keywords

Trajectory recovery, Pre-trained language model, Road network

\*Both authors contributed equally to this research.

†Corresponding author.

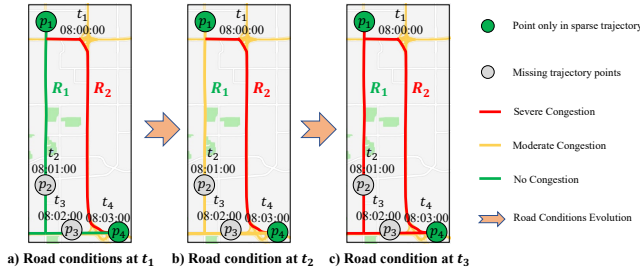
## 1 Introduction

A trajectory is a sequence of timestamped locations that describe the movement of individuals or vehicles, represented as  $\mathcal{T} = \langle p_1, \dots, p_{|\mathcal{T}|} \rangle$  of length  $|\mathcal{T}|$ , where  $p_i = (lat_i, lng_i, t_i)$  denotes the latitude, longitude, and timestamp of the  $i$ -th point. The sampling interval of  $\mathcal{T}$  is  $\mu = t_i - t_{i-1}, \forall i \in \{2, \dots, |\mathcal{T}|\}$ . With the advances in GPS devices and geopositioning technologies, a massive amount of trajectory data has been collected. These data play a pivotal role in various applications, such as urban planning [29], traffic management [19, 28], and personalized location services [27, 36]. However, factors such as network instability, device malfunctions, or cost-saving settings often lead to missing trajectory points, making the data sparse with large sampling intervals [20, 35]. Sparse trajectories fail to accurately reflect movement behavior and route choices, limiting their usefulness. To address this issue, recovering the missing points in sparse trajectories is crucial for preserving trajectory completeness—a task usually referred to as *trajectory recovery*.

Many works have been proposed to tackle trajectory recovery, broadly categorized into *free space trajectory recovery* [5, 7] and *map-matched trajectory recovery* [6, 30]. Free space methods directly predict the coordinates of missing points but do not ensure alignment with the road network, typically requiring a subsequent map-matching step. Map-matched methods, by contrast, aim to recover missing points directly on the road network, making them more convenient for downstream applications. Despite the progress of existing methods, map-matched trajectory recovery still faces the following challenges:

**1) Limited dense trajectory data.** Trajectory recovery models are data-driven and typically require large-scale pairs of sparse and dense trajectories. However, because of cost-saving settings and device malfunctions, most collected trajectories remain sparse. The limited availability of dense trajectory data makes existing models prone to overfitting and hampers their performance.

**2) Difficulty in generalizing to varying sampling intervals.** Due to network instability and device malfunctions, a sparse trajectory dataset often contains trajectories with a mixture of different



**Figure 1: Impact of road conditions on route selection and movement patterns.**

sampling intervals [28]. Existing works [6, 20, 30] usually treat all intervals in the same way, overlooking the varying spatiotemporal correlations of sparse trajectories entangled with multiple sampling intervals [2, 4]. As a result, when encountering trajectories with unseen sampling intervals, these models often require retraining, which increases costs.

**3) Non-trivial extraction of dynamic road conditions for missing points.** Road conditions are key to understanding movement patterns. Existing work [30] incorporates road conditions around observed points but ignores those at missing points. For example, in Figure 1(b) and (c), knowing the conditions at  $p_2$  and  $p_3$  reveals that the user is gradually decelerating on  $R_1$  due to congestion. If we only consider conditions at  $p_1$  and  $p_4$ , we might infer that the user takes  $R_1$  instead of  $R_2$  (Figure 1(a)), but not the finer details of the movement. Since the exact locations of missing points are unknown, their road conditions cannot be directly extracted.

To tackle these challenges, we propose *Pre-trained Language Model for Trajectory Recovery (PLMTrajRec)*. We leverage a pre-trained language model (PLM) that has already learned useful representations from large-scale corpus data and fine-tune it using only a small number of dense trajectories. This enables PLMTrajRec to recover missing points even when dense trajectories are scarce (*Challenge 1*). To handle sparse trajectories with varying sampling intervals (*Challenge 2*), we introduce an interval and feature-guided (IF-guided) *explicit trajectory prompt*. It encodes both sampling intervals and movement features into the PLM through natural language, helping the model identify these intervals and capture key trajectory characteristics. We then propose an *interval-aware trajectory embedder* to standardize different sampling intervals and learn their spatiotemporal correlations. To infer road conditions for missing points (*Challenge 3*), we design an area flow-guided (AF-guided) *implicit trajectory prompt* that gathers traffic flows in each region. We also present a *road condition passing mechanism* that uses road conditions from nearby observed points to estimate those of the missing points.

Overall, the main contributions of our work can be summarized as follows:

- We introduce PLMTrajRec, a trajectory recovery model with strong scalability. By leveraging a pre-trained language model, PLMTrajRec can be fine-tuned on a small set of dense trajectories while still achieving effective recovery.
- We propose an IF-guided explicit trajectory prompt and an interval-aware trajectory embedder to help PLMTrajRec generalize to

sparse trajectories with various sampling intervals. The prompt encodes intervals and movement features into the PLM, and the embedder unifies intervals and learns spatiotemporal correlations.

- We design an AF-guided implicit trajectory prompt to capture road conditions by aggregating traffic flows, and a road condition passing mechanism to infer missing-point conditions from nearby observations.
- We conduct extensive experiments on two real-world datasets, each with three sampling intervals, showing that PLMTrajRec achieves state-of-the-art performance in effectiveness, scalability, and generalizability.

## 2 Related Work

**Free-space Trajectory Recovery.** Trajectory recovery in free space aims to directly restore missing GPS coordinates. Early methods relied on predefined rules to model mobile objects [3, 22], such as the vehicle moving with uniform linear motion [11] or taking the most popular route [7]. Other approaches [1, 37] use Markov models to capture spatial transitions. However, these methods are limited to modeling low-order transitions and fail to capture the global spatial-temporal dependencies essential for accurate trajectory recovery. Recent deep learning-based methods have provided more effective solutions [32, 33]. DHTR [26] employs a GRU-based seq2seq model to analyze user route transitions, integrating Kalman filtering to improve accuracy. AttnMove [31] leverages attention mechanisms and LSTM to capture movement preferences. PeriodicMove [24] models trajectories as graphs using GNNs to extract user patterns. TERI [5] employs a transformer-based two-stage framework for irregular interval trajectory recovery, and TrajBERT [21] adapts BERT [8] for implicit trajectory recovery at a coarser granularity. Despite their effectiveness, these methods require an additional map-matching step to align recovered trajectories with road networks before they can be applied in navigation systems. This two-stage process leads to error accumulation and significant time overhead due to the computational cost of map-matching.

**Map-matched Trajectory Recovery.** Compared with the above methods, map-matched trajectory recovery incorporates the road network as input and aims to recover the trajectory directly onto the road. To this end, MTrajRec [20] first represents the trajectory point with a road segment and moving rate, utilizing the seq2seq-based multi-task framework to capture the spatiotemporal correlations within the trajectory. Following this idea, RNTrajRec [6] models the relationship between trajectories and road networks, designing a spatial-temporal transformer architecture to encode sparse trajectories. LightTR [16] proposes a lightweight trajectory recovery framework based on federated learning. MM-STGED [30] models sparse trajectories from a graph perspective, considering both micro and macro semantic information of trajectories. Although they demonstrate promising progress in trajectory recovery, their performance is still hindered by the poor availability of large-scale dense trajectories and limited in generalization to different sampling intervals discussed in Section 1.

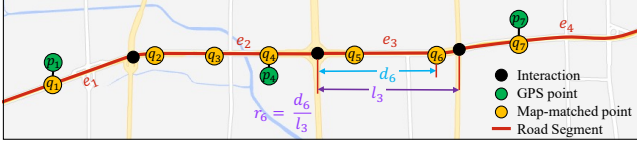


Figure 2: An illustration of map-matched trajectory, road segment  $e$ , and moving ratio  $r$ .

### 3 Preliminaries

#### 3.1 Definitions

**Definition 3.1 (Trajectory).** A trajectory is defined as a series of timestamped locations, denoted as  $\mathcal{T} = \langle p_1, \dots, p_{|\mathcal{T}|} \rangle$  where  $p_i = (lat_i, lng_i, t_i)$  represents the latitude and longitude coordinates of an object at the time  $t_i$ ,  $i \in \{1, \dots, |\mathcal{T}|\}$ .  $|\mathcal{T}|$  is the length of trajectory. The sampling interval of the trajectory  $\mathcal{T}$  is  $t_i - t_{i-1}$ ,  $i \in \{2, \dots, |\mathcal{T}|\}$ .

**Definition 3.2 (Road Network).** A road network is defined as a directed graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  denotes the set of nodes. Each node  $v \in \mathcal{V}$  signifies an intersection that links various road segments, and each node possesses attributes of latitude and longitude.  $\mathcal{E}$  represents the set of edges, where each edge  $e \in \mathcal{E}$  denotes road segment that connecting two intersections. An edge can be characterized by its starting intersection  $e.start \in \mathcal{V}$  and ending intersection  $e.end \in \mathcal{V}$ .

**Definition 3.3 (Map-matched Trajectory).** By employing the map-matching algorithm, a trajectory  $\mathcal{T}$  can be projected onto the road network and return the map-matched trajectory  $\mathcal{T}_m$ . As a result, each point of the map-matched trajectory aligns accurately with a particular road. The map-matched trajectory is represented as  $\mathcal{T}_m = \langle q_1, \dots, q_{|\mathcal{T}_m|} \rangle$ , where each  $q_j = (e_j, r_j, t_j)$  signifies the vehicle's location at time  $t_j$ ,  $j \in \{1, \dots, |\mathcal{T}_m|\}$ . Here,  $e \in \mathcal{E}$  is the matched road segment, and  $r$  is the moving ratio, which quantifies the proportion of the distance traveled from the starting point of a road segment in relation to the total length of that road segment.

**Example 3.4.** Figure 2 gives an example of a map-matched trajectory. For a sparse trajectory  $\mathcal{T}_s = \langle p_1, p_4, p_7 \rangle$  with the sampling interval of 90 seconds, its corresponding dense map-matched trajectory is represented by  $\mathcal{T}_m = \langle q_1, \dots, q_7 \rangle$  with a sampling interval of 30 seconds. Take a matching point  $q_6$  as an example. It falls on the road  $e_3$ , so its road segment is  $e_3$ . The distance of  $q_6$  from the start point of the road is represented as  $d_6$ , while the total length of  $e_3$  is denoted by  $l_3$ . Therefore, the moving ratio  $r_6$  can be calculated as  $r_6 = \frac{d_6}{l_3}$ .

#### 3.2 Problem Statement

**Map-matched Trajectory Recovery.** Given a sparse trajectory  $\mathcal{T}_s = \langle p_1, \dots, p_{|\mathcal{T}_s|} \rangle$  with a sampling interval of  $\mu$ . The goal of map-matched trajectory recovery is to reconstruct the dense map-matched trajectory  $\mathcal{T}_m = \langle q_1, \dots, q_{|\mathcal{T}_m|} \rangle$  with a sampling interval of  $\epsilon$ . Note that the sampling interval  $\mu$  is larger than  $\epsilon$ .

## 4 Methodology

### 4.1 Overall Pipeline

In this paper, we present both scalable and generalizable trajectory recovery model, PLMTrajRec, by fine-tuning a PLM that is pre-trained on a large-scale corpus with limited dense trajectory data. The framework of PLMTrajRec, shown in Figure 3, comprises three main components: dual trajectory prompts, an interval-aware trajectory embedder, and a PLM-based trajectory encoder.

The dual trajectory prompts provide essential information through two key components. First, the interval and feature-guided (IF-guided) explicit trajectory prompt, expressed in natural language, injects the sampling interval of sparse trajectories and their movement features into PLM. This enhances the model's ability to recognize the sampling intervals of each sparse trajectory and capture trajectory characteristics. Second, area flow-guided (AF-guided) implicit trajectory prompts model road conditions, offering crucial contextual information to improve the recovery of missing trajectory points.

The interval-aware trajectory embedder first normalizes sparse trajectories with varying sampling intervals  $\mu$  into a unified interval  $\epsilon$ , effectively handling the diverse spatiotemporal correlations associated with different sampling intervals and enhancing model generalization. Subsequently, each trajectory point, including both observed and missing points, is encoded into an embedding that the PLM can process.

In the PLM-based trajectory encoder, pre-trained BERT is employed to capture information bi-directionally, and most of the parameters are frozen to preserve the pre-trained knowledge, while the multi-head attention layer remains trainable, allowing the model to capture the spatiotemporal correlations within the trajectory. Finally, the output layer predicts the road segment  $e$  and the moving ratio  $r$  for the recovered map-matched trajectory at each step.

### 4.2 Dual Trajectory Prompts

To enable PLM to recover trajectories with different intervals and effectively model road conditions for missing trajectory points, we introduce dual trajectory prompts, including an interval and feature-guided (IF-guided) explicit trajectory prompt and an area flow-guided (AF-guided) implicit trajectory prompt.

**4.2.1 IF-guided Explicit Trajectory Prompt.** The IF-guided explicit trajectory prompt offers a structured textual description of the sampling intervals and movement features of sparse trajectories, enabling PLMTrajRec to identify varying sampling intervals and capture trajectory features. The prompts related to the sampling interval include the following components:

- **<Task Part>**: Sparse trajectory recovery.
- **<Target Part>**: Output the road segment and moving ratio for each point in the trajectory.
- **<Content Part>**: The sparse trajectory is sampled every  $\{large\}$  sampling interval and aims to recover trajectory every  $\{small\}$  sampling interval seconds.

The content within the placeholders  $\{\}$  is filled with trajectory-specific information. All trajectories share the same interval-related prompt. The **<Task Part>** informs the PLM about the overall task to be performed, the **<Target Part>** defines the required output format,

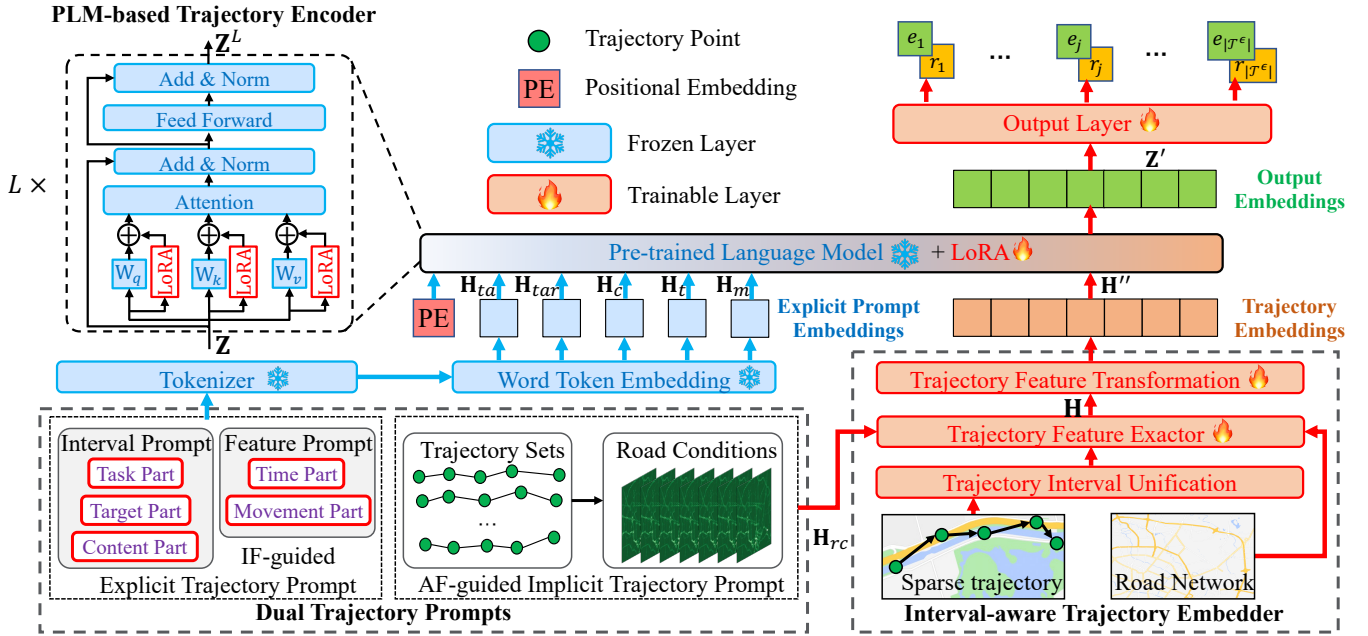


Figure 3: The framework of PLMTrajRec, consists of Dual Trajectory Prompts, Interval-aware Trajectory Embedder, and PLM-based Trajectory Encoder.

and the <Content Part> specifies the sampling intervals, guiding the PLM in effectively analyzing the trajectories.

The prompts related to the movement features include the following contents:

- **<Time Part>**: The trajectory started at {start time} on {day-in-week} and ended at {end time} on {day-in-week}.
- **<Movement Part>**: Total time cost: {x minutes y seconds}. Total space transfer distance: {z kilometers}.

The <Time Part> provides the trajectory’s specific start and end times, helping the PLM understand the duration and potential time patterns, such as morning or evening peaks. The <Movement Part> supports the PLM in inferring the trajectory’s movement. We give a detailed example of the IF-guided explicit trajectory prompt in Appendix C.

After obtaining the prompt for each part, we use the tokenizer and word token embedding in PLMs to convert text into embeddings. The embedding of <Task Part>, <Target Part>, <Content Part>, <Time Part>, and <Movement Part> are denoted by  $\mathbf{H}_{ta}$ ,  $\mathbf{H}_{tar}$ ,  $\mathbf{H}_c$ ,  $\mathbf{H}_t$ , and  $\mathbf{H}_m$ . These embeddings are then concatenated to form the overall IF-guided explicit trajectory prompt embedding.

$$\mathbf{H}^e = \mathbf{H}_{ta} || \mathbf{H}_{tar} || \mathbf{H}_c || \mathbf{H}_t || \mathbf{H}_m, \quad (1)$$

where  $||$  is the concatenate operation.

**4.2.2 AF-guided Implicit Trajectory Prompt.** Road conditions can provide insights into both the surrounding environment and the object’s movement, which are helpful for trajectory recovery. For instance, vehicles typically slow down in congested areas and accelerate in less congested regions. Given the complexity and variability of real-world road conditions are difficult to express in natural language, we represent them as the implicit trajectory prompt.

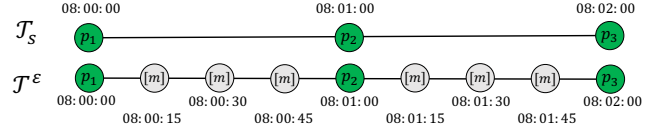


Figure 4: An illustration of trajectory preprocessing layer.

To obtain the road conditions for each trajectory point, we first calculate the average road conditions across all areas and time intervals, then extract the relevant information for each point. Specifically, we divide the area of interest into a spatial grid of  $I \times J$  cells and partition the time into  $T$  slices. For each grid cell, we compute the total traffic, resulting in a regional flow matrix  $\mathbf{RC} \in \mathbb{R}^{I \times J \times T}$ . Each element in  $\mathbf{RC}$  represents the number of traffic flows at region  $i, j$  at time  $t$ . To capture the spatiotemporal correlations among different regions, we apply a 2D convolution (2D CNN) along the spatial dimension and a 1D CNN along the temporal dimension, yielding the road condition representation  $\mathbf{H}_{rc} \in \mathbb{R}^{I \times J \times T \times F}$ , where  $F$  is the number of features. Formally:

$$\mathbf{H}_{rc} = \text{Conv1d}(\text{Conv2d}(\mathbf{RC})) \in \mathbb{R}^{I \times J \times T \times F} \quad (2)$$

Obviously, extracting the road conditions of a trajectory point needs its longitude, latitude, and timestamp. However, it is challenging for missing points as their geographic coordinates are unknown. Inspired by the message passing mechanism [9], we introduce a *road condition passing mechanism* that approximates the road conditions of missing points by leveraging the surrounding road conditions. The implementation details will be elaborated in Section 4.3.2.

### 4.3 Interval-aware Trajectory Embedder

To make the model deal with sparse trajectories with different sampling intervals and capture the spatiotemporal correlation, we propose an interval-aware trajectory embedder.

**4.3.1 Trajectory Interval Unification.** For different sparse trajectories, their sampling intervals  $\mu$  are not constant, such as 4 minutes, 2 minutes, or 1 minute, which bring different spatiotemporal correlations between trajectory points. To this end, we convert these into the same sampling interval with the target trajectory by introducing a placeholder '[m]' to mark missing trajectory points and generate a preprocessed sparse trajectory  $\mathcal{T}^\epsilon$ , where the length of  $\mathcal{T}^\epsilon$  is  $\frac{p[\mathcal{T}_s].t - p_1.t}{\epsilon} + 1$ . Although the location of '[m]' is unknown, its timestamp can still be calculated, as illustrated in Figure 4.

**4.3.2 Trajectory Feature Extractor.** Given the preprocessed sparse trajectory  $\mathcal{T}^\epsilon$ , there are two cases for extract trajectory point features:

- Case 1: the trajectory point  $s \in \mathcal{T}^\epsilon$  is observed. i.e.,  $\exists k \in \{1, \dots, |\mathcal{T}_s|\}, p_k.t = s.t$ .
- Case 2: the location of trajectory point  $s \in \mathcal{T}^\epsilon$  is missing. i.e.,  $s = [m]$ , note that its time  $s.t$  is known.

For case 1, we use Learnable Fourier Features (LFF) [15, 25] to encode the continuous latitude and longitude of  $s$  into an  $F$ -dimensional vector using the feature mapping function  $\Phi(x) : \mathbb{R} \rightarrow \mathbb{R}^F$ . Formally:

$$\Phi(x) = W_\Phi [\cos x W_r \parallel \sin x W_r], \quad (3)$$

where  $W_r \in \mathbb{R}^{F/2}$  and  $W_\Phi \in \mathbb{R}^{F \times F}$  represent the learnable parameters, and  $x \in \{s.lat, s.lon\}$ . Using the feature mapping function  $\Phi(\cdot)$ , the relative information  $x - y$  between points  $x$  and  $y$  can be captured through multiplication operations, which is important for understanding movement, such as distance. We provide detailed insight and proof of LFF in Appendix D.

Additionally, considering that vehicles move within the road network, the relationship between a trajectory point and its surrounding road segments is critical. To measure the relationship between a trajectory point  $s$  and a road segment  $l$ , we define a function  $f(d_{s,l})$  based on their shortest distance:

$$f(d_{s,l}) = \begin{cases} e^{-(\frac{d_{s,l}}{\kappa})^2} & \text{if } d_{s,l} < \varphi_{dist}, \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

where  $d_{s,l}$  is the shortest distance between  $s$  and road  $l$ ,  $\kappa$  is a hyperparameter, and  $\varphi_{dist}$  is a distance threshold. Then, we obtain the road network representation  $\mathbf{h}_s^{\text{road}}$  by:

$$\mathbf{h}_s^{\text{road}} = \frac{\sum_{l=1}^{|\mathcal{V}|} f(d_{s,l}) \cdot \mathbf{M}_l}{\sum_{l=1}^{|\mathcal{V}|} f(d_{s,l})}, \quad (5)$$

where  $\mathbf{M}_l \in \mathbb{R}^F$  is the learnable embedding of road segment  $l$ , and  $|\mathcal{V}|$  is the total number of road segments. Finally, the complete representation of a trajectory point  $s$  is obtained as:

$$\mathbf{h}_s = W_1 [(\Phi(s.lat) + \Phi(s.lng)) \parallel \mathbf{h}_s^{\text{road}}] + b_1, \quad (6)$$

where  $W_1 \in \mathbb{R}^{F \times 2F}$  and  $b_1 \in \mathbb{R}^F$  is the learnable parameters.

For case 2, where the location of point  $s$  is unknown, here we use road conditions to represent its features, as described in Section 4.2.2. Since the road conditions at a specific point are influenced by the surrounding conditions, which propagate along both temporal and spatial dimensions, we propose a road condition passing mechanism to infer the road conditions of the missing location inspired by the message passing mechanism [9].

**Road Condition Passing Mechanism.** First, for the missing point  $s$ , we identify the observed forward and backward trajectory points,  $s_f$  and  $s_b$ , respectively. For instance, in Figure 4, suppose we have a missing point  $s$  timestamped at 8:00:45, its observed forward point  $s_f = p_1$  and the backward point  $s_b = p_2$ . Then, we retrieve the road conditions of  $s_f$  and  $s_b$  as follows:

$$\begin{aligned} \mathbf{h}_{rc}^f &= \mathbf{H}_{rc} [\pi_{lat}(s_f.lat), \pi_{lng}(s_f.lng), \pi_t(s_f.t)], \\ \mathbf{h}_{rc}^b &= \mathbf{H}_{rc} [\pi_{lat}(s_b.lat), \pi_{lng}(s_b.lng), \pi_t(s_b.t)], \end{aligned} \quad (7)$$

where  $\pi_{lat}$ ,  $\pi_{lng}$ , and  $\pi_t$  are index functions mapping latitude, longitude, and time to their corresponding indices. Next, we calculate the time interval between  $s$  and  $s_f$  and  $s_b$ , denoted as  $\Delta t_f = s.t - s_f.t$  and  $\Delta t_b = s_b.t - s.t$ . The road condition of point  $s$  is then computed as:

$$\mathbf{h}_{rc}^s = \frac{e^{-\Delta t_f} \mathbf{h}_{rc}^f + e^{-\Delta t_b} \mathbf{h}_{rc}^b}{e^{-\Delta t_f} + e^{-\Delta t_b}} \quad (8)$$

In addition, we encode the time intervals  $\Delta t_f$  and  $\Delta t_b$  to quantify the relative position of  $s$ . The final representation of  $s$  is given by:

$$\mathbf{h}_s = W_2 [\mathbf{m} \parallel \text{FC}(\Delta t_f \parallel \Delta t_b) \parallel \mathbf{h}_{rc}^s] + b_2, \quad (9)$$

where  $\mathbf{m} \in \mathbb{R}^F$  is a learnable vector to represent the location is missing, and  $\text{FC}(\cdot) : \mathbb{R}^2 \rightarrow \mathbb{R}^F$  is the fully connection layer.  $W_2 \in \mathbb{R}^{F \times 3F}$  and  $b_2 \in \mathbb{R}^F$  are the learnable parameters. Finally, the overall trajectory representation is  $\mathbf{H} \in \mathbb{R}^{|\mathcal{T}^\epsilon| \times F}$ .

**4.3.3 Trajectory Feature Transformation.** To enhance the model's ability to comprehend trajectory features, we introduce  $K$  reference tokens  $\mathbf{E}_w \in \mathbb{R}^{K \times F}$ , inspired by [10, 13], to bridge the connection between PLMs and the trajectory. These reference tokens are designed to capture the global semantics of the trajectory.

Given the trajectory embedding  $\mathbf{H}$ , we first apply a 1D CNN to aggregate neighboring information and capture local movement patterns:

$$\mathbf{H}' = \text{Conv1d}(\mathbf{H}) \quad (10)$$

Next, we compute self-attention between the trajectory embedding  $\mathbf{H}'$  and the reference tokens  $\mathbf{E}_w$  to capture the global semantics, where  $\mathbf{H}'$  acts as the query, and  $\mathbf{E}_w$  as both the key and value:

$$\mathbf{H}'' = \text{Attention}(\mathbf{H}', \mathbf{E}_w, \mathbf{E}_w) \quad (11)$$

After transforming the embedding  $\mathbf{H}''$  in the token space, we concatenate it with the explicit trajectory prompt embedding  $\mathbf{H}^e$  to form the final trajectory representation:

$$\mathbf{Z} = \mathbf{H}^e \parallel \mathbf{H}'' \quad (12)$$

Then, we add the Transformer positional embedding  $PE$  in each element of  $\mathbf{Z}$  and feed it into PLMs encoder to encode trajectory.



#### 4.4 PLM-based Trajectory Encoder

We use the pre-trained BERT as the fundamental architecture of PLMs, considering that its encoder-only structure is well-suited to the reconstruction task due to its effective use of bi-directional contextual information from the trajectory [23]. To make PLM fully adaptable to the trajectory recovery task, we employ the Low-Rank Adaptation (LoRA) algorithm [12]. LoRA introduces additional parameters within the Transformer block, reducing the number of trainable parameters and computational complexity while enhancing the PLM’s performance on new tasks.

Specifically, in each self-attention block of the Transformer, we introduce additional weights  $\Delta \mathbf{W}_q \in \mathbb{R}^{F \times F}$ ,  $\Delta \mathbf{W}_k \in \mathbb{R}^{F \times F}$ , and  $\Delta \mathbf{W}_v \in \mathbb{R}^{F \times F}$  for the query, key, and value matrices  $\mathbf{W}_q$ ,  $\mathbf{W}_k$ , and  $\mathbf{W}_v$ . To reduce the parameter count, the additional weights  $\Delta \mathbf{W}$  are decomposed into two low-rank matrices,  $\mathbf{B} \in \mathbb{R}^{F \times r}$  and  $\mathbf{C} \in \mathbb{R}^{r \times F}$ , where  $r \ll F$ . The modified query, key, and value matrices in each self-attention block are updated as  $\mathbf{W}_q = \mathbf{W}_q + \Delta \mathbf{W}_q$ ,  $\mathbf{W}_k = \mathbf{W}_k + \Delta \mathbf{W}_k$ , and  $\mathbf{W}_v = \mathbf{W}_v + \Delta \mathbf{W}_v$ . Formally, the output of the  $l$ -th transformer layer is defined as:

$$\mathbf{Z}^l = \text{LoRA}(\text{Transformer}(\mathbf{Z}^{l-1})) \quad (13)$$

After stacking  $L$  Transformer layers, we discard the IF-guided explicit trajectory prompt portion and obtain the output embeddings  $\mathbf{Z}' \in \mathbb{R}^{|\mathcal{T}^\epsilon| \times d}$ . For the  $j$ -th element of  $\mathbf{Z}'$ , we apply the softmax function to calculate the probability of each road segment and use the argmax function to return the predicted road segment  $e_j$ . We employ a MLP with a Sigmoid activation function to determine the moving ratio  $r_j$ .

#### 4.5 Training

**4.5.1 Loss Function.** We use multi-task learning to optimize both road segment recovery and moving ratio recovery simultaneously. For the road segment, we use the cross-entropy loss function:

$$\mathcal{L}_e = -\frac{1}{|\mathcal{T}^\epsilon|} \sum_{j=1}^{|\mathcal{T}^\epsilon|} \log(\hat{e}_j | \mathbf{Z}'_j) \quad (14)$$

For the moving ratio, we use the mean squared error loss function:

$$\mathcal{L}_r = \frac{1}{|\mathcal{T}^\epsilon|} \sum_{j=1}^{|\mathcal{T}^\epsilon|} |r_j - \hat{r}_j|^2 \quad (15)$$

The final loss function is  $\mathcal{L} = \mathcal{L}_e + \lambda \mathcal{L}_r$ , where  $\lambda$  is a hyperparameter to balance two tasks.

**4.5.2 Joint Training Strategy.** To generalize PLMTrajRec in recovering sparse trajectories with various sampling intervals, we propose the following training approach.

For the dense trajectory dataset  $\mathbb{T}$ , which is set as a 15-second sampling interval in our experiments. To create sparse trajectory datasets, we re-sample each trajectory  $\mathcal{T} \in \mathbb{T}$  at three different intervals: 1 minute, 2 minutes, and 4 minutes, resulting in three datasets:  $\mathbb{T}_1$ ,  $\mathbb{T}_2$ , and  $\mathbb{T}_4$ . These are then combined to form a larger training dataset  $\mathbb{T}_{all}$ . Formally,

$$\mathbb{T}_{all} = \mathbb{T}_1 \cup \mathbb{T}_2 \cup \mathbb{T}_4. \quad (16)$$

By training PLMTrajRec on  $\mathbb{T}_{all}$ , we achieve balanced performance across the different sampling intervals. Subsequently, we

**Table 1: Dataset Description.**

Types	Chengdu	Porto
Sampling interval	15s	15s
# Trajectory	118,354	322,079
# Road segment	2504	2224
Latitude range	30.655 ~ 30.727	41.142 ~ 41.174
Longitude range	104.043 ~ 104.129	-8.652 ~ -8.578
Length / Width	8.22 km / 8.00 km	6.19 km / 3.56 km

fine-tune the model on each individual sparse trajectory dataset to further improve trajectory recovery performance.

### 5 Experiments

In this section, we present comprehensive experiments to evaluate the effectiveness, scalability, and generalization capabilities of PLM-TrajRec on two real-world trajectory datasets with three different sparse trajectory sampling intervals.

#### 5.1 Datasets

We assess our model using two publicly available trajectory datasets from Chengdu, China, and Porto, Portugal. The Chengdu dataset<sup>1</sup> contains trajectory data collected in November 2016. The Porto dataset<sup>2</sup> includes trajectory from 442 taxis, collected from July 2013 to June 2014. Both datasets are standardized to a sampling interval of 15 seconds. We remove trajectories with travel times less than 5 minutes or exceeding 1 hour, as well as outlier trajectories. The road network data is obtained from the OpenStreetMap website<sup>3</sup>. We employ a map-matching algorithm [17] to project the trajectory onto the road network and obtain the ground truth of the road segments and moving rates. The detailed overview of the dataset characteristics is provided in Table 1.

#### 5.2 Baselines

To evaluate the effectiveness of our model, we compare PLMTrajRec with 12 baseline methods. These include five free space trajectory recovery models: **HMM** [17] + **ShortestPath**, **Linear** [11] + **HMM** [17], **MPR** [7] + **HMM** [17], **DHTR** [26] + **HMM** [17], and **AttnMove** [31] + **Rule**, as well as seven map-matched trajectory recovery models: **MTrajRec** [20], **T2vec** [14] + **Decoder**, **T3s** [34] + **Decoder**, **TERI** [5] + **Decoder**, **TrajBERT** [21] + **Decoder**, **RN-TrajRec** [6], and **MM-STGED** [30]. Due to space constraints, the details of each baseline are provided in Appendix A.

#### 5.3 Evaluation Metrics

We employ five common metrics to assess the effectiveness of our model followed [6, 20, 30]. For road segment recovery, we utilize **Accuracy (Acc)**, **Recall**, and **Precision (Prec)** metrics to evaluate. And employ **Mean Absolute Error (MAE)** and the **Root Mean Square Error (RMSE)** in assessing the recovered GPS coordinates. The details of each metric are provided in Appendix B.

<sup>1</sup><https://outreach.didichuxing.com/>

<sup>2</sup><https://www.kaggle.com/competitions/pkdd-15-predict-taxi-service-trajectory-i/data>

<sup>3</sup><http://www.openstreetmap.org/>

**Table 2: Performance comparison on two datasets with sampling intervals at 4 minutes, 2 minutes, and 1 minute, respectively. The best results are highlighted in bold, while the underline indicates the second-best results.**

Sampling Interval	Methods	Chengdu					Porto				
		Acc(%)	Recall(%)	Prec(%)	MAE	RMSE	Acc(%)	Recall(%)	Prec(%)	MAE	RMSE
$\mu = 4$ minutes $\rightarrow \epsilon = 15$ seconds	HMM + ShortestPath	26.85	28.64	29.55	939.3	1047.7	20.19	26.22	33.51	886.9	941.5
	Linear + HMM	26.42	30.45	36.15	974.5	1145.4	32.23	36.09	49.80	489.3	637.3
	MPR + HMM	36.93	38.62	44.53	821.9	914.1	32.22	38.67	48.07	534.0	700.2
	DHTR + HMM	41.48	57.34	50.48	673.6	911.3	32.02	58.29	45.61	456.7	627.1
	AttnMove + Rule	63.43	73.97	78.72	358.2	916.7	49.31	48.62	78.03	310.0	621.3
	MTrajRec	65.79	75.14	78.42	315.1	904.4	52.36	<u>60.39</u>	77.28	266.1	590.1
	T3s + Decoder	65.60	75.26	78.14	318.2	926.3	52.24	60.24	77.80	270.4	594.9
	T2vec + Decoder	66.51	75.68	78.27	307.5	915.2	53.13	60.27	77.62	256.3	571.0
	TERI + Decoder	66.42	75.59	78.36	309.2	903.8	53.59	60.02	78.26	253.9	558.3
	TrajBERT + Decoder	66.09	75.38	78.59	310.7	911.4	52.98	60.12	77.93	251.8	560.1
	RNTrajRec	67.66	75.59	79.97	306.1	886.0	54.59	<b>60.42</b>	79.20	248.1	549.1
	MM-STGED	70.64	76.04	81.63	266.2	829.7	57.30	<b>59.48</b>	80.21	222.8	510.4
	<b>PLMTrajRec</b>	74.12	79.63	86.46	262.8	483.0	57.61	59.15	<b>82.19</b>	200.9	376.9
	<b>PLMTrajRec + FT</b>	<b>74.58</b>	<b>80.09</b>	<b>86.63</b>	<b>253.2</b>	<b>465.7</b>	<b>57.67</b>	59.08	<u>82.05</u>	<b>200.8</b>	<b>370.2</b>
$\mu = 2$ minutes $\rightarrow \epsilon = 15$ seconds	HMM + ShortestPath	33.85	47.89	48.31	754.1	826.2	27.30	40.05	46.00	647.3	747.0
	Linear + HMM	43.78	45.35	48.77	816.9	1054.7	49.35	50.45	63.87	408.7	609.8
	MPR + HMM	49.88	54.62	50.94	474.8	899.0	49.76	52.97	62.86	409.8	610.9
	DHTR + HMM	47.17	60.16	51.73	662.0	912.2	43.76	65.25	52.10	385.3	578.0
	AttnMove + Rule	71.98	77.42	80.67	291.1	764.8	61.39	60.90	82.98	213.4	468.5
	MTrajRec	74.52	78.25	81.09	254.5	885.7	61.65	65.65	78.99	179.9	451.5
	T3s + Decoder	74.62	78.95	81.79	242.2	857.5	61.75	65.53	79.14	181.3	461.2
	T2vec + Decoder	75.69	78.86	81.68	231.6	783.6	62.24	65.77	78.97	173.9	438.0
	TERI + Decoder	75.32	78.74	81.38	239.0	823.7	62.14	65.39	79.18	178.3	443.9
	TrajBERT + Decoder	75.20	78.69	81.53	235.0	813.7	62.34	65.66	79.02	177.9	441.2
	RNTrajRec	75.80	79.35	81.86	218.5	757.0	63.39	65.84	79.25	171.3	433.9
	MM-STGED	78.14	80.06	83.58	197.2	696.0	65.69	66.15	80.74	152.5	400.8
	<b>PLMTrajRec</b>	81.76	84.31	88.38	187.4	366.2	66.40	<u>66.52</u>	82.14	141.9	294.6
	<b>PLMTrajRec + FT</b>	<b>82.29</b>	<b>84.59</b>	<b>88.72</b>	<b>181.2</b>	<b>349.0</b>	<b>66.72</b>	<b>66.87</b>	<b>82.38</b>	<b>141.7</b>	<b>293.5</b>
$\mu = 1$ minute $\rightarrow \epsilon = 15$ seconds	HMM + ShortestPath	35.92	67.92	60.16	529.7	638.2	34.75	48.46	48.67	527.2	659.3
	Linear + HMM	68.59	65.66	66.67	707.4	1005.2	66.17	64.72	75.22	368.3	571.1
	MPR + HMM	62.25	62.67	60.53	418.8	659.0	66.27	65.66	74.51	402.6	628.2
	DHTR + HMM	51.09	63.40	50.14	584.7	750.4	52.98	69.60	57.17	420.4	625.8
	AttnMove + Rule	79.60	81.55	82.75	194.4	752.6	72.07	69.59	80.41	156.6	360.7
	MTrajRec	81.12	81.73	83.75	187.1	718.4	71.65	70.92	80.35	114.9	332.3
	T3s + Decoder	80.90	82.78	83.15	187.1	713.0	71.78	71.61	80.16	110.1	328.0
	T2vec + Decoder	81.69	81.90	83.88	185.6	714.1	71.86	71.10	80.48	114.2	334.7
	TERI + Decoder	81.25	81.89	83.92	186.9	710.4	71.53	71.38	80.29	113.2	325.9
	TrajBERT + Decoder	81.38	81.72	83.65	183.2	710.7	71.63	71.47	80.39	112.7	329.5
	RNTrajRec	81.88	82.09	84.84	177.9	702.5	72.31	71.88	80.57	110.3	325.7
	MM-STGED	84.26	84.15	85.92	154.0	633.5	73.16	72.27	80.81	108.2	321.9
	<b>PLMTrajRec</b>	<u>87.17</u>	<u>87.99</u>	<u>90.52</u>	141.4	290.8	<u>74.42</u>	<u>72.78</u>	<u>82.82</u>	<u>95.6</u>	<b>211.7</b>
	<b>PLMTrajRec + FT</b>	<b>88.15</b>	<b>88.82</b>	<b>91.04</b>	<b>138.6</b>	<b>289.1</b>	<b>75.35</b>	<b>73.87</b>	<b>83.28</b>	<b>95.2</b>	<u>220.3</u>

## 5.4 Settings

We split the trajectory dataset into training, validation, and testing sets in a 7:2:1 ratio. Following previous work [6, 20, 30], for dense map-matched trajectories, we set the sampling interval  $\epsilon = 15$  seconds. To generate sparse trajectories, for each dense trajectory, we retain the first and last points and create three sparse versions by setting the sampling intervals  $\mu = 4$  minutes, 2 minutes, and 1 minute, respectively.

During training, we integrate sparse trajectories with three different sampling intervals to train PLMTrajRec. After training, we evaluate its performance in recovering trajectories across various sampling intervals to assess its generalizability. To further enhance performance, we fine-tune the trained model for each specific sampling interval, allowing for a more precise recovery.

We employ PyTorch [18] framework to implement PLMTrajRec, with a learning rate of  $1e-4$  and a batch size of 64. BERT-small<sup>4</sup> is selected as the foundation model for PLM with 4 transformer layers and the number of hidden state is 512. For road condition

<sup>4</sup><https://github.com/google-research/bert>

extraction, the area of interest is divided into a  $64 \times 64$  grid, and the time dimension is partitioned into hourly intervals. The hidden state dimension is set to  $F = 512$ . In Equation 4, we set  $\kappa = 15$  and  $\varphi_{dist} = 50$  meters. The model is trained for 50 epochs with early stopping, using a patience of 10 epochs. All experiments are conducted on NVIDIA RTX A4000 GPUs.

## 5.5 Experimental Results

Table 2 reports a comparison of the results between our model and the baselines across various sampling intervals of sparse trajectories on both the Chengdu and Porto datasets. As the sampling interval increases, the map-matched trajectory recovery task is more difficult. Due to the more complex road network structure in Porto compared to Chengdu, with Porto having 100.9 roads /  $km^2$  and Chengdu having 38.1 roads /  $km^2$ , the accuracy of road segment recovery is lesser in Porto than in Chengdu.

Compared to other baselines, our model achieves superior performance across all metrics, with an average improvement of 16.51% on the Chengdu dataset and 9.35% on the Porto dataset. Notably, it

**Table 3: Scalability analysis. The performance comparison on the Chengdu dataset when trained with different data ratios.**

Sampling Interval	Data Ratio	20%		40%		60%		80%		100%	
	Metric	Acc(%)	RMSE	Acc(%)	RMSE	Acc(%)	RMSE	Acc(%)	RMSE	Acc(%)	RMSE
$\mu = 2$ minutes $\rightarrow \epsilon = 15$ seconds	MTrajRec	69.85	919.8	72.58	907.4	73.97	902.3	74.32	891.3	74.52	885.7
	T3s + Decoder	68.86	909.5	72.29	897.2	73.63	883.8	74.53	868.8	74.62	857.5
	T2vec + Decoder	69.47	905.4	72.77	858.3	73.80	855.8	74.89	831.3	75.69	783.6
	RNTrajRec	68.05	991.0	70.20	856.9	71.17	834.4	72.62	795.3	75.80	757.0
	MM-STGED	71.65	865.0	75.32	773.0	75.49	752.1	76.94	747.7	78.14	696.0
	<b>PLMTrajRec</b>	<b>76.29</b>	<b>452.8</b>	<b>79.37</b>	<b>411.5</b>	<b>80.37</b>	<b>395.4</b>	<b>81.18</b>	<b>377.4</b>	<b>81.76</b>	<b>366.2</b>

**Table 4: Zero-shot study on Chengdu dataset with sampling intervals at 2 minutes.**

Methods	Acc(%)	MAE	RMSE
MTrajRec	56.39 (↓ 18.13%)	492.7 (↓ 48.37%)	1046.2 (↓ 15.34%)
T3s + Decoder	63.81 (↓ 10.81%)	353.0 (↓ 31.39%)	980.6 (↓ 9.68%)
T2vec + Decoder	63.32 (↓ 12.37%)	376.6 (↓ 38.50%)	928.1 (↓ 15.57%)
RNTrajRec	64.93 (↓ 10.87%)	339.6 (↓ 35.70%)	912.8 (↓ 17.07%)
MM-STGED	68.27 (↓ 9.90%)	291.8 (↓ 32.42%)	862.6 (↓ 19.31%)
<b>PLMTrajRec</b>	<b>79.62 (↓ 2.14%)</b>	<b>195.3 (↓ 4.05%)</b>	<b>379.6 (↓ 3.53%)</b>

shows substantial gains in RMSE metrics, with an average reduction of 351.8 meters in Chengdu and 119.2 meters in Porto. This indicates PLMTrajRec generalizes effectively, enabling it to accurately recover trajectories with varying sampling intervals. By fine-tuning at the specific sampling interval, **PLMTrajRec + FT**, our model further improves Acc by 1.72% and 0.32%. These outstanding results can be attributed to the strong generalization capability of the PLM in processing sparse trajectories with varying sampling intervals. Additionally, the interval-aware trajectory embedder plays a pivotal role to capture spatial-temporal correlations and converting trajectory embeddings into a format that the PLM can effectively process and understand.

## 5.6 Scalability Analysis

To evaluate the effectiveness of PLMTrajRec in trajectory recovery when dense trajectory data is limited, we conduct scalability experiments on the Chengdu dataset. Specifically, we train the model using subsets of the training set at 20%, 40%, 60%, 80%, and 100%, and evaluate performance on the test set. The results are presented in Table 3 and 7 of Appendix F.2. As the size of the training set increases, we observe performance improvements across all baseline models. Notably, with only 20% of the training data, PLMTrajRec already surpasses most baselines, trailing only MM-STGED. With 40% of the training data, PLMTrajRec outperforms all other models, demonstrating its strong scalability. This advantage underscores the practicality of PLMTrajRec in real-world scenarios, where obtaining large-scale dense trajectory data is often challenging.

## 5.7 Zero-shot Study on Sampling Interval

We further analyze the generalizability of our model when the target sampling interval is absent during training. Specifically, PLMTrajRec is trained with a mixture of sampling intervals of 1 minute and 4 minutes and tested on a 2-minute sampling interval. The experimental results on the Chengdu dataset are shown in Table 4, where

**Table 5: Ablation study on Chengdu dataset with sampling intervals at 2 minutes.**

Methods	Acc(%)	Recall(%)	Prec(%)	MAE	RMSE
w/o IF-guided explicit trajectory prompt	81.04	84.08	88.07	205.8	394.2
w/o AF-guided implicit trajectory prompt	81.43	84.28	88.14	191.4	371.1
w/o dual trajectory prompts	80.70	83.61	87.73	225.9	407.4
w/o road network	81.28	84.22	88.04	194.5	375.6
w/o reference tokens	80.83	83.36	86.68	236.6	437.2
PLMTrajRec - Randomly initialized BERT	74.11	78.62	80.83	288.4	831.9
<b>PLMTrajRec</b>	<b>81.76</b>	<b>84.31</b>	<b>88.38</b>	<b>187.4</b>	<b>366.2</b>

↓ indicates the percentage decline compared to training with a 2-minute sampling interval. We observe that the accuracy of all models decreased when the target sampling interval was not included in the training data. However, our model shows a notably smaller performance drop. This robustness is attributed to the trajectory prompts and joint training strategy used by PLMTrajRec, which effectively identify the sampling interval of the target trajectories, demonstrating strong generalizability. In contrast, baseline models rely solely on trajectory point information to capture correlations, which struggles with generalization due to the inconsistency of temporal dynamics between the training and testing.

## 5.8 Ablation Study

To evaluate the effectiveness of the proposed components, we create six variants of PLMTrajRec. **w/o IF-guided explicit trajectory prompt, w/o AF-guided implicit trajectory prompt, w/o dual trajectory prompt, w/o road network, w/o reference tokens, and PLMTrajRec - Randomly initialized BERT**. The detailed description of each variant is shown in Appendix E.

As shown in Table 5 and 8, each component of PLMTrajRec is essential. The IF-guided explicit trajectory prompt provides task and trajectory-related textual information, enabling PLMs to effectively understand trajectory features. Omitting it leads to a performance decrease. When the AF-guided implicit trajectory prompt is removed, many missing locations are replaced by the '[MASK]' token in BERT, limiting the model's ability to capture road conditions for missing points, which results in suboptimal performance. For the trajectory point embedder, excluding road network for observed points causes the model to rely solely on latitude and longitude, neglecting the relationship between trajectories and the road network, which demonstrates that road network is crucial for modeling trajectory information. Additionally, removing reference tokens results in a performance decline, as PLMs are unable to directly interpret trajectory data. Using the randomly initialized BERT in PLMTrajRec has poor performance, indicating the efficiency of the PLM for trajectory recovery.



## 6 Conclusion

In this paper, we propose PLMTrajRec, a novel model for trajectory recovery leveraging PLM. PLMTrajRec effectively recovers trajectory even with limited dense trajectory data available, demonstrating strong scalability. The model equippes with an interval and feature-guided explicit trajectory prompt and interval-aware trajectory prompt enabling it to effectively generalize to different sampling intervals. Additionally, we introduce an area flow-guided implicit trajectory prompt to gather traffic flows in each region, and propose a road condition passing mechanism to infer missing-point conditions from nearby observations. Experimental results on two datasets with three different sampling intervals validate the effectiveness, scalability, and generalizability of the proposed model.

## References

- [1] Prithu Banerjee, Sayan Ranu, and Sriram Raghavan. 2014. Inferring uncertain trajectories from partial observations. In *2014 IEEE International Conference on Data Mining*. IEEE, 30–39.
- [2] Zhengping Che, Sanjay Purushotham, Guangyu Li, Bo Jiang, and Yan Liu. 2018. Hierarchical deep generative models for multi-rate multivariate time series. In *International Conference on Machine Learning*. PMLR, 784–793.
- [3] Dawei Chen, Cheng Soon Ong, and Lexing Xie. 2016. Learning points and routes to recommend trajectories. In *Proceedings of the 25th ACM international conference on information and knowledge management*. 2227–2232.
- [4] Jiawei Chen, Pengyu Song, and Chunhui Zhao. 2024. Multi-scale self-supervised representation learning with temporal alignment for multi-rate time series modeling. *Pattern Recognition* 145 (2024), 109943.
- [5] Yile Chen, Gao Cong, and Cuauhtemoc Anda. 2023. Teri: An effective framework for trajectory recovery with irregular time intervals. *Proceedings of the VLDB Endowment* 17, 3 (2023), 414–426.
- [6] Yuqi Chen, Hanyuan Zhang, Weiwei Sun, and Baihua Zheng. 2023. Rntrajrec: Road network enhanced trajectory recovery with spatial-temporal transformer. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 829–842.
- [7] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. 2011. Discovering popular routes from trajectories. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 900–911.
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.
- [10] Shengnan Guo, Youfang Lin, Letian Gong, Chenyu Wang, Zeyu Zhou, Zekai Shen, Yiheng Huang, and Huaiyu Wan. 2023. Self-supervised spatial-temporal bottleneck attentive network for efficient long-term traffic forecasting. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1585–1596.
- [11] Sahar Hoteit, Stefano Secchi, Stanislav Sobolevsky, Carlo Ratti, and Guy Pujolle. 2014. Estimating human trajectories and hotspots through mobile phone data. *Computer Networks* 64 (2014), 296–307.
- [12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).
- [13] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*. PMLR, 3744–3753.
- [14] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE, 617–628.
- [15] Yang Li, Si Si, Gang Li, Cho-Jui Hsieh, and Samy Bengio. 2021. Learnable fourier features for multi-dimensional spatial positional encoding. *Advances in Neural Information Processing Systems* 34 (2021), 15816–15829.
- [16] Ziqiao Liu, Hao Miao, Yan Zhao, Chenxi Liu, Kai Zheng, and Huan Li. 2024. LightTR: A Lightweight Framework for Federated Trajectory Recovery. *arXiv preprint arXiv:2405.03409* (2024).
- [17] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 336–343.
- [18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
- [19] Zeenat Rehena and Marijn Janssen. 2018. Towards a framework for context-aware intelligent traffic management system in smart cities. In *Companion Proceedings of the The Web Conference 2018*. 893–898.
- [20] Huimin Ren, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li, and Yu Zheng. 2021. Mtrajrec: Map-constrained trajectory recovery via seq2seq multi-task learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1410–1419.
- [21] Junjun Si, Jin Yang, Yang Xiang, Hanqiu Wang, Li Li, Rongqing Zhang, Bo Tu, and Xiangqun Chen. 2023. TrajBERT: BERT-Based Trajectory Recovery with Spatial-Temporal Refinement for Implicit Sparse Trajectories. *IEEE Transactions on Mobile Computing* (2023).
- [22] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. 2013. Calibrating trajectory data for similarity-based analysis. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*. 833–844.
- [23] Chenxi Sun, Yaliang Li, Hongyan Li, and Shenda Hong. 2023. TEST: Text prototype aligned embedding to activate LLM’s ability for time series. *arXiv preprint arXiv:2308.08241* (2023).
- [24] Hao Sun, Changjie Yang, Liwei Deng, Fan Zhou, Feiteng Huang, and Kai Zheng. 2021. Periodicmove: Shift-aware human mobility recovery with graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1734–1743.
- [25] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems* 33 (2020), 7537–7547.
- [26] Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, and Kai Feng. 2019. Deep trajectory recovery with fine-grained calibration using kalman filter. *IEEE Transactions on Knowledge and Data Engineering* 33, 3 (2019), 921–934.
- [27] Qinyong Wang, Hongzhi Yin, Tong Chen, Zi Huang, Hao Wang, Yanchang Zhao, and Nguyen Quoc Viet Hung. 2020. Next point-of-interest recommendation on resource-constrained mobile devices. In *Proceedings of the Web conference 2020*. 906–916.
- [28] Sheng Wang, Zhifeng Bao, J Shane Culpepper, and Gao Cong. 2021. A survey on trajectory data management, analytics, and learning. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–36.
- [29] Yu Wang, Tongya Zheng, Yuxuan Liang, Shunyu Liu, and Mingli Song. 2024. Cola: Cross-city mobility transformer for human trajectory simulation. In *Proceedings of the ACM on Web Conference 2024*. 3509–3520.
- [30] Tonglong Wei, Youfang Lin, Yan Lin, Shengnan Guo, Lan Zhang, and Huaiyu Wan. 2024. Micro-Macro Spatial-Temporal Graph-Based Encoder-Decoder for Map-Constrained Trajectory Recovery. *IEEE Transactions on Knowledge and Data Engineering* (2024).
- [31] Tong Xia, Yunhan Qi, Jie Feng, Fengli Xu, Funing Sun, Diansheng Guo, and Yong Li. 2021. Attnmove: History enhanced trajectory recovery via attentional network. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4494–4502.
- [32] Fengli Xu, Zhen Tu, Yong Li, Pengyu Zhang, Xiaoming Fu, and Depeng Jin. 2017. Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data. In *Proceedings of the 26th international conference on world wide web*. 1241–1250.
- [33] Shuai Xu, Donghai Guan, Zhuo Ma, and Qing Meng. 2022. A Temporal-Context-Aware Approach for Individual Human Mobility Inference Based on Sparse Trajectory Data. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Springer, 106–120.
- [34] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3s: Effective representation learning for trajectory similarity computation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2183–2188.
- [35] Kai Zhao, Jie Feng, Zhao Xu, Tong Xia, Lin Chen, Funing Sun, Diansheng Guo, Depeng Jin, and Yong Li. 2019. DeepMM: Deep learning based map matching with data augmentation. In *Proceedings of the 27th ACM SIGSPATIAL international conference on advances in geographic information systems*. 452–455.
- [36] Pengpeng Zhao, Anjing Luo, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. 2020. Where to go next: A spatio-temporal gated network for next poi recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2020), 2512–2524.
- [37] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. 2012. Reducing uncertainty of low-sampling-rate trajectories. In *2012 IEEE 28th international conference on data engineering*. IEEE, 1144–1155.

## A Baseline Setting

We choose the following 12 methods as baselines, including five free space trajectory recovery and seven map-matched trajectory recovery.

### A.1 Free-space Trajectory Recovery

Free space trajectory recovery first recovers trajectory points and then projects the trajectory onto the road network.

- **HMM [17] + ShortestPath** first projects the sparse trajectory on the road network based on the hidden markov model (HMM), and then calculate the shortest path.
- **Linear [11] + HMM [17]** linearly interpolates missing trajectory points and then implements HMM to perform the map matching process.
- **MPR [7] + HMM [17]** first divides the area of interest into grids to identify frequently traveled routes between sparse trajectory points. Then, it assumes that vehicle movement maintains a constant speed for trajectory recovery. Subsequently, it employs HMM to project the trajectory onto the road network.
- **DHTR [26] + HMM [17]** incorporates a sequence-to-sequence framework and Kalman filtering to recover trajectory points. Subsequently, it applies an HMM to yield a trajectory that is constrained by the map.
- **AttnMove [31] + Rule** uses attention to predict the missing road segments and uses the central location as the moving rate.

### A.2 Map-matched Trajectory Recovery

Map-matched trajectory recovery can directly recover the trajectory on the road network.

- **MTrajRec [20]** utilizes a sequence-to-sequence framework with Gated Recurrent Units (GRU) as the key component for trajectory recovery. It optimizes road segment and moving rate prediction through multi-task learning.
- **T2vec [14]** is a deep learning model for trajectory similarity learning. We use its encoder to embed the sparse trajectory.
- **T3s [34]** uses LSTM and attention mechanisms to encode sparse trajectory data effectively.
- **TERI [5]** assumes that the number of missing trajectory points is unknown and proposes a two-stage trajectory recovery framework. In the first stage, a transformer-based model predicts the number of points to be recovered, and in the second stage, the same framework is used for recovery trajectory coordinates. Here, we utilize only the second stage of TERI.
- **TrajBERT [21]** employs a transformer encoder and a forward and backward neighbor selector to learn complex mobility patterns bi-directionally from sparse trajectories.
- **RNTrajRec [6]** leverages the Transformer to capture the spatial-temporal correlation of the sparse trajectories. It also takes into account the relation between the trajectory and the road network.
- **MM-STGED [30]** models sparse trajectories from a graph perspective and recovers trajectories by capturing micro and macro semantic information.

Notably, despite T2vec, T3S, TERI, and TrajBERT employing different techniques to capture sparse trajectories' spatial-temporal dependencies, they cannot directly generate the desired format for

missing points. Therefore, after these models obtain the trajectory embeddings, we append the Decoder part of MTrajRec to output the road segment and moving ratio of the trajectory point. Denoted by **T2v + Decoder**, **T3S + Decoder**, **TERI + Decoder**, and **TrajBERT + Decoder**, respectively.

## B Evaluation Metrics

We adopt five widely used metrics to evaluate the effectiveness of our model, following previous works [6, 20, 30]. For road segment recovery, we use **Accuracy (Acc)**, **Recall**, and **Precision (Prec)** to assess the alignment between the true road segments  $\mathcal{E}_p = \{e_1, \dots, e_m\}$  and the predicted road segments  $\hat{\mathcal{E}}_p = \{\hat{e}_1, \dots, \hat{e}_m\}$ . A higher value in these metrics indicates a more accurate road segment recovery. The metrics are formally defined as follows:

$$\begin{aligned} \text{Acc} &= \frac{1}{m} \sum_{i=1}^m \mathbb{1}\{e_i = \hat{e}_i\} \times 100\%, \\ \text{Recall} &= \frac{|\mathcal{E}_p \cap \hat{\mathcal{E}}_p|}{|\hat{\mathcal{E}}_p|} \times 100\%, \\ \text{Prec} &= \frac{|\mathcal{E}_p \cap \hat{\mathcal{E}}_p|}{|\mathcal{E}_p|} \times 100\%, \end{aligned} \quad (17)$$

where  $\mathbb{1}\{\cdot\}$  is the indicator function, where  $e_i = \hat{e}_i$ ,  $\mathbb{1}\{e_i = \hat{e}_i\} = 1$ , otherwise  $\mathbb{1}\{e_i = \hat{e}_i\} = 0$ .

To evaluate the recovered GPS coordinates, we employ the **Mean Absolute Error (MAE)** and **Root Mean Square Error (RMSE)** to quantify the distance error between the true trajectory  $\mathcal{T}_m = q_1, \dots, q_{|\mathcal{T}_m|}$  and the predicted trajectory  $\hat{\mathcal{T}}_m = \hat{q}_1, \dots, \hat{q}_{|\hat{\mathcal{T}}_m|}$ . The formulas for MAE and RMSE are given as follows: The formulas of MAE and RMSE are as follows:

$$\begin{aligned} \text{MAE} &= \frac{1}{|\mathcal{T}_m|} \sum_{i=1}^{|\mathcal{T}_m|} |\text{RN\_dist}(q_i, \hat{q}_i)|, \\ \text{RMSE} &= \sqrt{\frac{1}{|\mathcal{T}_m|} \sum_{i=1}^{|\mathcal{T}_m|} |\text{RN\_dist}(q_i, \hat{q}_i)|^2} \end{aligned} \quad (18)$$

Here, following [6, 20, 30],  $\text{RN\_dist}(q, \hat{q})$  signifies the shortest distance along the road network between the trajectory points  $q$  and  $\hat{q}$ . Both MAE and RMSE are denoted in meters. Lower values of these metrics indicate a higher level of accuracy in the recovery results.

## C Example of the IF-guided Explicit Trajectory Prompt

Consider a sparse trajectory  $\mathcal{T} = \langle p_1, \dots, p_N \rangle$  of  $N$  trajectory points with a sampling interval of 4 minutes, starting at 8 o'clock on Saturday and ending at 9 o'clock on Saturday. Our goal is to recover it within a sampling interval of 15 seconds. Therefore, the trajectory prompts that are related to sampling intervals are: **Task Part**: Sparse trajectory recovery. **Target Part**: Output the road segment and moving ratio for each point in the trajectory. **Content Part**: The sparse trajectory is sampled every four minutes and aims to recover trajectory every fifteen seconds. The movement feature-related trajectory prompts are: **Time Part**: The trajectory started at eight o'clock on Saturday and ended at nine o'clock on Saturday. **Movement Part**: Total time cost: sixty minutes zero seconds. Total

**Table 6: Hyperparameter range and optimal value.**

Parameter	Range
The number of reference tokens $K$	128, 256, <u>512</u> , 1024
LoRA rank $r$	4, <u>8</u> , 16, 64
The loss function weight $\lambda$	0.1, 1, <u>10</u> , 100

space transfer distance:  $z$  kilometers. Here  $z = \sum_{i=2}^N \text{dist}(p_i, p_{i-1})$ , where  $\text{dist}(\cdot, \cdot)$  is used to calculate the distance between two trajectory points.

## D Insight about the Learnable Fourier Features

Consider two trajectory points  $x$  and  $y$ , and the feature mapping function  $\Phi(x) = W_\Phi[\cos xW_r \parallel \sin xW_r]$  in Learnable Fourier Features. The relative information  $x - y$  between points  $x$  and  $y$  can be captured through multiplication operations, i.e.:

$$\begin{aligned}
 \Phi(x) \cdot \Phi(y) &= W_\Phi[\cos xW_r \parallel \sin xW_r] \cdot W_\Phi[\cos yW_r \parallel \sin yW_r] \\
 &= \|W_\Phi\|_2 \cdot (\cos x \cdot \cos y + \sin x \cdot \sin y) \cdot \|W_r\|_2 \\
 &= \|W_\Phi\|_2 \cdot \cos(x - y) \cdot \|W_r\|_2
 \end{aligned} \tag{19}$$

In our PLMTrajRec, after feature conversion, we input the trajectory feature into a pre-language training model based on BERT. Since there are a lot of multiplication-based attention operations in the PLM, the relative information  $x - y$  can be easily modeled and utilized. This relative information helps infer crucial details like the distance between trajectory points, which is important for understanding movement. For instance, a larger distance between two points may indicate a higher likelihood of vehicle acceleration.

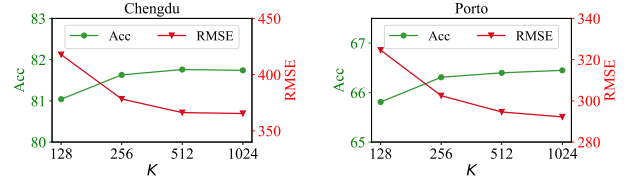
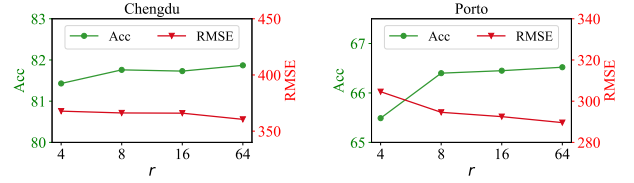
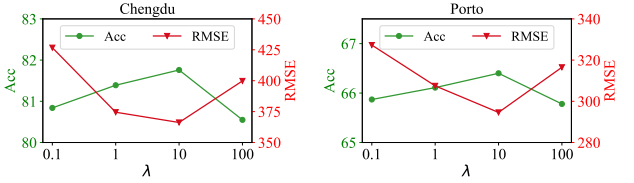
## E The Detailed Description of Variant

- **w/o IF-guided explicit trajectory prompt:** We remove the IF-guided explicit trajectory prompt.
- **w/o AF-guided implicit trajectory prompt:** We use the '[MASK]' token in the BERT to represent the missing location.
- **w/o dual trajectory prompts:** We remove both the IF-guided explicit and implicit trajectory prompt, and the missing points are represented by the '[MASK]' token in BERT.
- **w/o road network:** We only use the Learnable Fourier Feature for encoding the observed trajectory points while removing the road network feature.
- **w/o reference tokens:** We remove the trajectory feature transformation layer in the trajectory embedder module.
- **PLMTrajRec - Randomly initialized BERT:** We randomly initialize the parameter of BERT instead of using the pre-trained BERT based on large-scale corpus datasets.

## F Appendix Experiments

### F.1 Hyperparameter Study

To explore the impact of hyperparameters on model performance, we conduct hyperparameter analysis using the Chengdu and Porto datasets with the sparse trajectory sampling interval of 2 minutes.


**Figure 5: Hyperparameter analysis of the number of reference tokens  $K$  of trajectory feature transformation.**

**Figure 6: Hyperparameter analysis of the rank  $r$  of LoRA.**

**Figure 7: Hyperparameter analysis of the weight  $\lambda$  in the loss function.**

We consider three important hyperparameters, including the number of reference tokens  $K$ , the rank  $r$  of LoRA, and the loss function weight  $\lambda$ . Details regarding the range of selection and the optimal values for these hyperparameters are presented in Table 6.

**F.1.1 The Number of Reference Tokens  $K$ .** We set the parameter  $K$  from the set  $\{128, 256, 512, 1024\}$  to explore its impact on trajectory recovery. The experimental results are shown in Figure 5. As  $K$  increases, the effectiveness of trajectory recovery also improves. This suggests that having a larger token space aids in accurately representing trajectory features. When  $K$  is larger than 512, the performance improvement of PLMTrajRec becomes marginal, while the computational cost will increase. To balance the performance and efficiency of the model, we set  $K$  to 512.

**F.1.2 The Rank  $r$  of LoRA.** As shown in Figure 6, we set the range of  $r$  to 4, 8, 16, and 64 to explore its impact. It is observed that the model performs well when  $r = 4$ . As  $r$  increases, the accuracy will continue to increase, but it is not obvious. This suggests that PLMs encapsulate substantial domain expertise through training on extensive corpora, rendering them adaptable to trajectory recovery tasks with minor adjustments. Yet as  $r$  increases, the number of parameters also increases, making the model training require more

**Table 7: Scalability analysis on Chengdu dataset with 1 minute and 4 minutes sampling intervals.**

Sampling Interval	Data Ratio Metric	20%		40%		60%		80%		100%	
		Acc(%)	RMSE	Acc(%)	RMSE	Acc(%)	RMSE	Acc(%)	RMSE	Acc(%)	RMSE
$\mu = 4$ minutes $\rightarrow \epsilon = 15$ seconds	MTrajRec	60.73	1048.0	63.58	968.5	64.73	929.4	65.39	914.6	65.79	904.4
	T3s + Decoder	60.49	1098.9	63.32	968.7	64.23	961.9	65.04	942.3	65.60	926.3
	T2vec + Decoder	62.17	966.1	64.92	946.1	64.95	972.3	65.31	946.8	66.51	915.2
	RNTrajRec	60.80	998.0	62.85	931.3	64.19	909.1	65.46	835.0	67.66	886.0
	MM-STGED	64.61	935.0	67.76	881.8	68.53	853.7	69.95	825.3	70.64	829.7
	<b>PLMTrajRec</b>	<b>68.30</b>	<b>585.3</b>	<b>71.57</b>	<b>515.6</b>	<b>72.74</b>	<b>512.5</b>	<b>73.56</b>	<b>488.3</b>	<b>74.12</b>	<b>483.0</b>
$\mu = 1$ minute $\rightarrow \epsilon = 15$ seconds	MTrajRec	75.39	937.5	78.53	835.1	80.01	794.3	80.93	725.7	81.12	718.4
	T3s + Decoder	76.49	917.1	79.08	824.9	80.65	767.8	80.82	716.5	80.90	713.0
	T2vec + Decoder	75.89	845.4	79.09	746.8	79.41	752.3	81.21	742.2	81.69	714.1
	RNTrajRec	75.65	846.2	79.48	784.8	79.61	769.3	81.74	742.5	81.88	702.5
	MM-STGED	76.02	857.7	79.86	734.7	82.25	676.4	83.44	663.0	84.26	633.5
	<b>PLMTrajRec</b>	<b>81.62</b>	<b>397.5</b>	<b>84.59</b>	<b>352.0</b>	<b>85.75</b>	<b>328.7</b>	<b>86.73</b>	<b>305.4</b>	<b>87.17</b>	<b>290.8</b>

**Table 8: Ablation study on Porto dataset with sampling intervals at 2 minutes.**

Methods	Acc(%)	Recall(%)	Prec(%)	MAE	RMSE
w/o IF-guided explicit trajectory prompt	66.13	66.19	81.98	157.9	314.6
w/o AF-guided implicit trajectory prompt	66.18	66.22	82.01	149.6	298.1
w/o dual trajectory prompts	65.52	65.89	81.51	171.0	339.4
w/o road network	66.11	66.27	81.90	150.7	302.8
w/o reference tokens	65.89	65.07	80.57	165.4	327.7
PLMTrajRec - Randomly initialized BERT	62.06	64.18	79.26	179.3	413.9
<b>PLMTrajRec</b>	<b>66.40</b>	<b>66.52</b>	<b>82.14</b>	<b>141.9</b>	<b>294.6</b>

**Table 9: Computation Cost.**

Dataset	Chengdu / Porto		
Methods	Model size (MBytes)	Train time (min/epoch)	Inference time (min)
DHTR	19 / 18	11.30 / 42.55	1.13 / 2.35
MTrajRec	49 / 44	37.19 / 142.12	12.57 / 28.20
RNTrajRec	112 / 109	72.30 / 153.29	29.14 / 43.58
MM-STGED	34 / 31	41.27 / 138.33	16.27 / 30.58
<b>PLMTrajRec</b>	<b>176 / 175</b>	<b>24.18 / 54.14</b>	<b>2.56 / 10.38</b>

memory. Therefore, we set  $r = 8$  to balance the performance and resources.

**F.1.3 The Loss Weight  $\lambda$ .** As shown in Figure 7, as the loss function weight  $\lambda$  increases, the model performance first improves and then decreases. This is because a smaller  $\lambda$  will make the model focus on road segment recovery, while a larger  $\lambda$  will focus on moving rate recovery. To balance these two tasks, we set the value of  $\lambda$  to 10.

## F.2 Scalability Study on 1-minute and 4-minute Sampling Interval

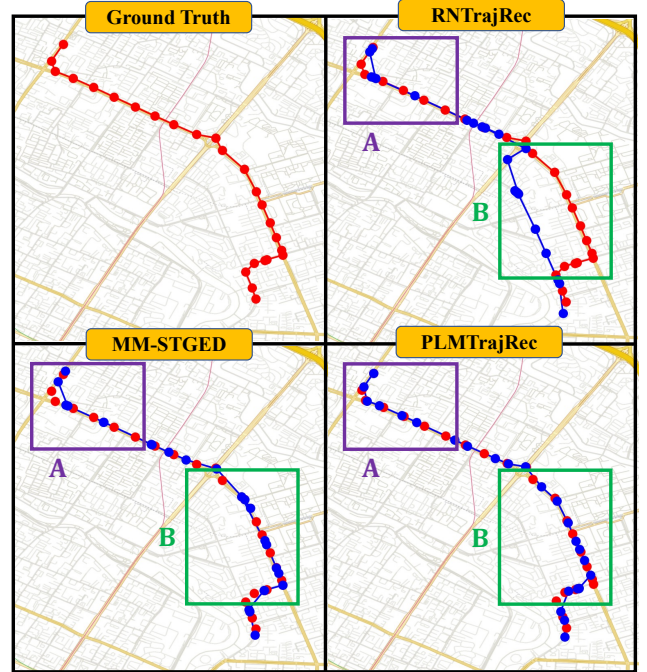
The result of the scalability study on the sparse trajectory with 1-minute and 4 minutes, as shown in Table 7, the conclusion is consistent with the 2-minute sampling interval. The experiment results prove the scalability of PLMTrajRec.

## F.3 Ablation Study on the Porto

The result of the ablation study on the Porto dataset, as shown in Table 8, the conclusion is consistent with the Chengdu dataset. Each component contributes to PLMTrajRec’s performance, and removing any component results in a decline in model effectiveness.

## F.4 Computational Cost

We use an NVIDIA RTX A4000 card to conduct cost analysis on the Chengdu and Porto datasets with a 2-minute sampling interval by considering model size, training time, and inference time. As shown in Table 9, compared to the end-to-end deep learning model, PLMTrajRec has the largest number of parameters due to its utilization of a pre-trained language model based on the BERT-small framework. Despite having a greater number of parameters, PLMTrajRec exhibits quicker training and testing times as it can output results for all trajectory points at once, and the majority of parameters are frozen, unlike other models that necessitate autoregressive generation. It significantly accelerates both model training and inference processes.



**Figure 8: Case study on the Chengdu dataset. Red points represent the truth trajectory points and blue points represent the recovered trajectory points.**

## F.5 Case Study

We conduct a case study on the Chengdu dataset to visualize the trajectory recovery performance with various baselines. As shown in Figure 8, we draw the truth and recovered trajectory, where red points represent the truth trajectory points and blue points indicate the recovered trajectory points. We find that the recovered trajectory aligns well with the road network, demonstrating the effectiveness of using road segment and moving rate to represent trajectory point. To facilitate a more intuitive comparison of recovery performance among different models, we focus on two distinct

regions, labeled as A and B, for visual analysis. In region A, characterized by a relatively simple road network structure, all models can accurately recover road segments. Among them, PLMTrajRec maps trajectory points better by leveraging PLMs. In contrast, region B exhibits a more intricate road network with multiple accessible routes. RNTrajRec captures spatial-temporal correlations of trajectories that are not adequate and recovers incorrect road segments. PLMTrajRec not only excels in road segment recovery but also in accurately matching the actual trajectory points. This success can be attributed to its ability to model missing trajectory points through the implicit trajectory prompt, thereby introducing more valuable information and improving performance.