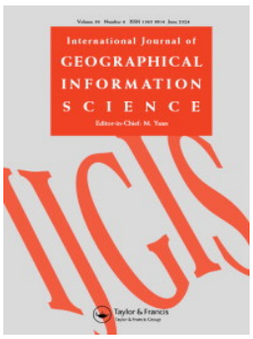

International Journal of Geographical Information Science

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tgis20

# Next track point prediction using a flexible strategy of subgraph learning on road networks

Yifan Zhang, Wenhao Yu & Di Zhu

**To cite this article:** Yifan Zhang, Wenhao Yu & Di Zhu (27 May 2024): Next track point prediction using a flexible strategy of subgraph learning on road networks, International Journal of Geographical Information Science, DOI: 10.1080/13658816.2024.2358527

**To link to this article:** https://doi.org/10.1080/13658816.2024.2358527

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
https://doi.org/10.1080/13658816.2024.2358527

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE



# Next track point prediction using a flexible strategy of subgraph learning on road networks

Yifan Zhang[a], Wenhao Yu[a,b] and Di Zhu[c]

[a]School of Geography and Information Engineering, China University of Geosciences, Wuhan, China; [b]National Engineering Research Center for Geographic Information System, China University of Geosciences, Wuhan, China; [c]Department of Geography, Environment and Society, University of Minnesota, Twin Cities, MN, USA

**ABSTRACT**

Accurately predicting the next track point of vehicle travel is crucial for various Intelligent Transportation System (ITS) applications, such as travel behavior studies, traffic control, and traffic congestion monitoring. Recent works on trajectory prediction follow a paradigm that first represents the raw trajectory and subsequently makes predictions based on that representation. Currently, trajectory representation methods tend to project trajectory points to road networks by map matching and represent trajectories based on the representation of matched roads. However, precisely matching trajectories to roads is a challenge in ITS, as the matching precision is greatly affected by the quality of the trajectory. Meanwhile, since it is difficult to discern whether trajectory matching results are accurate or confounded, how to effectively utilize this type of uncertain geographic context information is also a challenge, which is defined as the Uncertain Geographic Context Problem (UGCoP) in geographic information science. Therefore, we propose a flexible strategy of subgraph learning, referred to as SLM, for predicting the next track point of vehicles. Specifically, a subgraph generation module is first proposed to extract topology contextual information of the roads around historical trajectory points. Secondly, a subgraph learning module is designed to learn rich spatial and temporal features from generated subgraphs. Finally, the extracted spatiotemporal features will be fed into a prediction module to predict the next track points of vehicles on road networks. Our model enables the effective utilization of uncertain geographic context information of trajectories on road networks while avoiding the error brought by map matching. Extensive experiments based on trajectory datasets in two different cities confirm the effectiveness of our approach.

**ARTICLE HISTORY**
Received 26 September 2023
Accepted 19 May 2024

**KEYWORDS**
Trajectory representation; vehicle location prediction; uncertain geographic context problem; spatiotemporal dependency; intelligent transportation system

## 1. Introduction

With the prevalence of positioning services, such as the Global Positioning Service (GPS), an enormous amount of trajectory data has been collected. This has provided

CONTACT Wenhao Yu yuwh@cug.edu.cn

researchers with opportunities to explore and extract spatiotemporal patterns of human movement (Demšar and Virrantaus 2010; Siła-Nowicka *et al*. 2016), including predicting future locations (Huang *et al*. 2023; Liang *et al*. 2023). In Intelligent Transportation Systems (ITS), predicting the next locations of vehicles based on travel history is one of the most critical tasks (Chekol and Fufa 2022). It can be beneficial for various location-based services, such as autonomous driving (Bacciu *et al*. 2023; De Caro *et al*. 2023), navigation systems (Dai *et al*. 2015, Cui *et al*. 2018), location-based advertising (Ying *et al*. 2019), and traffic control (Bacciu *et al*. 2017; Chen *et al*. 2019a). For instance, by knowing where drivers will go next, traffic management authorities can dynamically adjust traffic signals to alleviate traffic congestion. Due to its practicality and importance, location prediction has garnered significant attention from a diverse range of fields, including transportation, computer science, and geography (Luca *et al*. 2021; Wu *et al*. 2018a).

Generally, most existing works on trajectory prediction follow a framework that first represents (or encodes) a raw trajectory and subsequently makes predictions based on that representation (Chen *et al*. 2021, Ebel *et al*. 2020, Kong and Wu 2018, Xu *et al*. 2021, Liu *et al*. 2016, Lin *et al*. 2021, Qian *et al*. 2019). Usually, travel routes of vehicles are measured by positioning services at fixed time intervals, and trajectory data are recorded in the form of sequences of locations (e.g., latitude and longitude). Intuitively, a straightforward way to encode trajectories is to use the coordinates of raw trajectory points as the input of prediction models (Figure 1b). However, the predicted locations based on only spatial coordinates will be largely influenced by spatial sparsity and noisy trajectory points (Li *et al*. 2018). To solve this problem, trajectory points are allocated into space grid cells, and grid cells are used to represent corresponding trajectory points (Figure 1c) (Xu *et al*. 2021). Despite this improvement, using grid cells as the spatial unit to represent trajectories ignores the constraint of road networks. In reality, the movement of vehicles relies heavily on transportation networks.

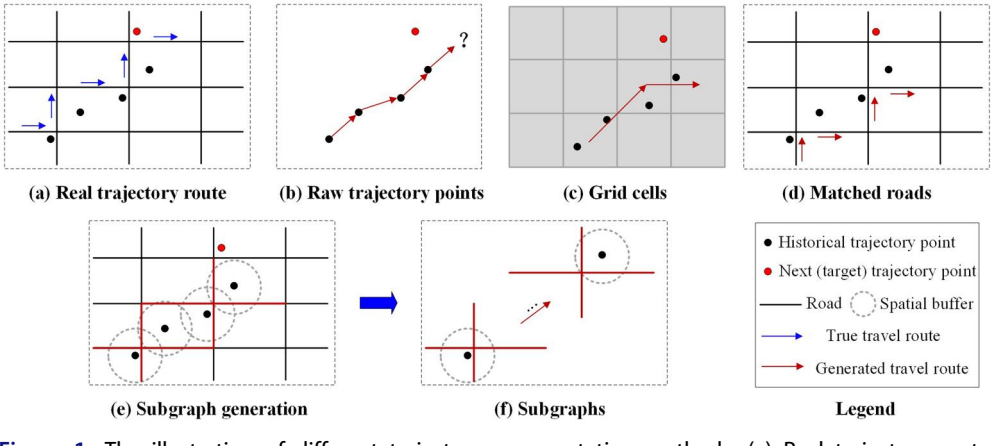


**Figure 1.** The illustration of different trajectory representation methods. (a) Real trajectory route, (b) raw trajectory points, (c) grid cells, (d) matched roads, and (e) our subgraphs: grey dotted line boxes denot spatial buffers around trajectory points and red lines denote extracted surrounding roads used to construct subgraphs in (f).

To effectively represent raw vehicles trajectory points, researchers attempted to introduce the constraint of road networks and proposed some solutions (Zhu and Liu 2017). For example, Wu *et al*. (2017) first projected trajectory points to road networks by map matching and subsequently used Recurrent Neural Network (RNN) to represent trajectories based on matched roads. Lin *et al*. (2021) also adopted map matching and designed an encoder-decoder framework to learn trajectory representation in a self-supervised form. There are many further works for trajectory representation (Qian *et al*. 2019, Mao *et al*. 2022, Liu *et al*. 2022, Chen *et al*. 2021b, Jiang *et al*. 2022), and a common paradigm of them is first to match trajectory points to road networks and subsequently represent trajectory based on the representation of matched roads (e.g., road embeddings). However, two issues are ignored in these works:

**The challenge of map matching.** Trajectory map matching is a challenge in ITS, and researchers have worked hard for a long time to solve this problem (Yang and Gidofalvi 2018, Huang *et al*. 2021, Wu *et al*. 2023). Moreover, it is difficult to obtain accurate map matching results when dealing with trajectories with low quality, such as low sampling rate and large measurement error (Li *et al*. 2021, 2023; Zhu and Liu 2017). An illustrated example is presented in Figure 1d, which shows that matching raw trajectory points to roads may obtain different results (red arrows) from the ground truth (blue arrows in Figure 1a). Therefore, inaccurate trajectory matching results will bring large errors to downstream location prediction tasks.

**The uncertain geographic context problem.** Due to the potential inaccuracies introduced by map matching algorithms, relying solely on matched roads as the representation of trajectory points for trajectory prediction overlooks the Uncertain Geographic Context Problem (UGCoP) (Kwan 2012b). UGCoP is a fundamental methodological problem that has garnered attention from researchers across multiple scientific disciplines, including geographic information science (Liu *et al*. 2023, Shmool *et al*. 2018, Robertson and Feick 2018), environmental science (Chen and Kwan 2015; Zhao *et al*. 2018), and social science (Kwan 2012a). Generally, UGCoP arises from spatial uncertainty in the areas influencing individuals and temporal uncertainty regarding the timing and duration of contextual influences (Kwan 2012b). Specifically, in trajectory prediction tasks, UGCoP (or to be exact the spatial uncertainty) arises when using matched roads as input geographic context, due to the limited knowledge of the precise spatial characteristics of the true geographic context. In other words, UGCoP in this context means it is difficult to discern whether matched road results are accurate or confounded, and if confounded, to what extent it may have distorted the final prediction results.

To solve these issues, we proposed a flexible strategy of subgraph learning, referred to as SLM (Subgraph Learning Model), for predicting the next track point of vehicles. In our method, trajectory matching is no longer needed and topology contextual information of road networks around trajectory points is flexibly utilized for next track point prediction. Specifically, a subgraph generation module initially creates a spatial buffer (grey dotted line boxes in Figure 1e) around historical trajectory points to extract surrounding topology contextual information on road networks (red lines in Figure 1e). The extracted information is employed to represent trajectory points, thereby mitigating uncertainty associated with representing trajectory points based on

matched roads through map matching. Subsequently, a subgraph learning module is designed to acquire rich spatial and sequential features from the generated subgraphs to assist the final prediction task. The prediction process is designed to output both predicted road IDs and corresponding movement ratios. This involves predicting both the road and the proportion (ranging from 0, indicating the vehicle is at the start of the specific road, to 1, indicating the vehicle is at the end) of the next track point. In this way, the input of SLM does not necessitate map matching, and it can also directly output corresponding location of vehicles on road networks without post-matching.

The main contribution of our study can be summarized as follows:

1. We propose a flexible strategy of subgraph learning called SLM on road networks for the next track point prediction of vehicles, which enables the effective utilization of uncertain geographic contextual information while avoiding the error brought by trajectory matching.
2. We design three major modules including subgraph generation, subgraph learning, and multi-task prediction. The former two modules can extract rich spatial and sequential features inherent in historical trajectories, while the last module can directly predict vehicle locations on road networks without post-matching.
3. We conduct extensive experiments based on trajectory datasets in two different cities to compare our model with recent methods. Experimental results demonstrate that our method significantly outperforms state-of-the-art solutions.

## 2. Related work

Trajectory prediction is a key problem in ITS, which has drawn attention for a long time, and researchers have actively explored various methods to tackle the issue (Huang *et al*. 2022; Rudenko *et al*. 2019). As one of the tasks in trajectory prediction, predicting the next track point of vehicles is also significantly important for urban traffic-related applications. For instance, predicting the positions of city vehicles in the next minute (or longer) allows urban traffic management authorities to better adjust signals, routes, and traffic flow, which can optimize smooth traffic and reduce congestion. In emergency situations, such as when ambulances or police cars require a rapid response, accurate prediction of the positions of other vehicles for the next minute and beyond can help them reach their destinations more quickly. Many researchers have focused on achieving more accurate predictions of the next position of vehicles. For example, Altch and Fortelle (2017) introduced LSTM (Hochreiter and Schmidhuber 1997) to predict the coordinates of vehicles on highways in the next time period (e.g., 1 second later). Rathore *et al*. (2019) proposed a scalable framework for predicting the next track point or long-term position of vehicles. Chen *et al*. (2021a) employed urban taxi trajectories to propose an origin-aware method for predicting the next track point of vehicles. In general, trajectory prediction and predicting the next track point of vehicles are focused by researchers in ITS (Wu *et al*. 2018b).

Traditionally, trajectory prediction is regarded as a statistical probabilistic problem, and some probabilistic models such as Markov models are proposed (Guo *et al*. 2018;

Schreier *et al*. 2014; Ye *et al*. 2016). With the advancement of artificial intelligence, the utilization of deep learning models for trajectory prediction has become a widely adopted solution. Among deep learning methods, a challenge is how to effectively capture sequential dependencies and geographical contextual information from historical trajectories. Given similar structural characters between trajectories and text (i.e., sequence), existing methods introduce relevant techniques from natural language processing (NLP) to model sequential dependencies of trajectories, such as RNN-based models (LSTM, GRU, and their variants) and the Transformer (Fu and Lee 2020, Chen *et al*. 2021b, Yang *et al*. 2021). As for geographical contextual information extraction, researchers tend to first learn effective trajectory representation methods in terms of geographic contexts, and subsequently extract contextual information within trajectories by encoding them based on learned representation. The geographical contextual information within trajectories of vehicles includes elements such as the topological structure of road networks (Wu *et al*. 2017, Liu *et al*. 2022, Lin *et al*. 2022), as well as road characteristics like type, length, number of lanes, and maximum travel speed (Jiang *et al*. 2022), to name a few.

Trajectory representation learning aims to embed complicated raw trajectory points into low-dimensional vectors. Effective trajectory representations can reveal implicit spatial information in trajectories, which is beneficial for various downstream tasks, such as trajectory prediction (Messaoud *et al*. 2021), trajectory classification (Kontopoulos *et al*. 2023, Chen *et al*. 2019c), and trajectory similarity computation (Han *et al*. 2021; Sousa *et al*. 2020). Related studies can be categorized based on the spatial carriers of trajectories, which are the spatial units used to carry and represent trajectory points (e.g., a grid cell or a road). First, some works represent trajectories based on geographical coordinates (i.e., longitude and latitude). For example, Yao *et al*. (2017) designed a sliding window to obtain space- and time- features in continuous coordinate sequences for representing trajectories. Though the strategy of using coordinates is intuitive, trajectory representation is highly influenced by spatial sparsity and noise of trajectory points. To solve this problem, t2vec (Li *et al*. 2018) proposed to partition the space into grid cells of equal size and treated each cell as a spatial carrier to represent trajectory points located in it. To further overcome the data sparsity problem, Ebel *et al*. (2020) proposed a k-d tree-based partitioning approach to partition space into grids of unequal sizes. Similarly, H-LATL (Xu *et al*. 2021) and NeuTraj (Yao *et al*. 2019) also used the grid as the spatial carrier to represent trajectories. Furthermore, T3S (Yang *et al*. 2021) proposed to integrate both spatial coordinates and grid to represent trajectories, which aimed to preserve the overall geographical distances between trajectories while capturing the structural information of trajectories. Moreover, some works also adopted geographical entities as spatial carriers to represent trajectories, such as areas of interest (Kong and Wu 2018, Chen *et al*. 2019b) and base stations (Lin *et al*. 2021; Wan *et al*. 2022).

Since we focus on the location prediction of vehicles in this paper while the movement of vehicles relies largely on road networks, using either coordinates or grid cells to represent trajectories overlooks the constraint of road networks. Therefore, trajectory representation considering road networks is important for modeling more accurate spatial contexts of vehicle trajectories. Some researchers have noticed this

problem and proposed some solutions. For example, DND (Qian *et al*. 2019) transformed a trajectory into a road junction sequence by map matching, and used the embeddings of road junctions to represent the trajectory. In this method, road segments are used as the spatial carrier of trajectories. Similarly, Trembr (Fu and Lee 2020) matched trajectory points to road networks and designed a recurrent neural network (RNN) based encoder-decoder model to train and learn trajectory representation. In this method, road representation is first learned based on a proposed Road2Vec model. Toast (Chen *et al*. 2021b) extended the skip-gram model from NLP by introducing auxiliary traffic context prediction for road network representation, and a trajectory-enhanced Transformer module is further designed for trajectory representation based on the learned road representation. Many trajectory representation learning methods in terms of road networks have also been proposed for specific optimization objects (Jiang *et al*. 2022; Liu *et al*. 2016; Wu *et al*. 2017). A common strategy of these methods is first matching trajectories to road networks, and subsequently learning trajectory representation based on the matched roads.

However, as described in Section 1, trajectory matching is also a challenge in ITS, and matching precision generally drops when noises exist in trajectories. Trajectory representation based on mismatched roads may bring large uncertainties to downstream tasks including location prediction. Moreover, information on neighboring roads around a trajectory point is also important to model its local geographical context. For example, considering topology information of neighboring roads around trajectory points can be helpful for trajectory recovery (Chen *et al*. 2022). Therefore, in this paper, we propose a flexible strategy of subgraph learning on road networks, which aims to represent trajectories based on neighboring roads for predicting the next locations of vehicles.

## 3. Preliminary

In this section, we formally introduce the key concepts related to this work and define the task of the next track point prediction of a vehicle.

**Definition 1: Road Network.** A road network is represented as a directed graph $G = (V, E)$, where $V$ represents the set of road segments and $E$ denotes the connectivity of these road segments.

For example, $E(v_i, v_j) = 1$ denotes there exists a direct connection from road segment $v_i$ to road segment $v_j$.

**Definition 2: Travel route and Trajectory.** A travel route $r = \{v_1, v_2, ..., v_{|r|}\}$ is a sequence of adjacent road segments, where $v_i \in V$, which represents a road segment. A raw trajectory $t = \{p_1, p_2, ..., p_{|t|}\}$ is a sequence of sampled points from the underlying route of a vehicle, which are recorded by positioning services with a time (sampling) interval, and each point $p_i$ corresponds to a coordinate of latitude and longitude, denotes $p_i = (lat_i, lng_i)$.

Positioning devices have measurement errors, thus a trajectory point usually locates around but not exactly on the travel route.

**Definition 3: Next track point prediction.** Given a road network $G = (V, E)$ and raw historical trajectory points of a vehicle $t = \{p_1, p_2, ..., p_{|t|}\}$, the task of next track point prediction aims to accurately forecast its next track point on the road network $\overrightarrow{p}_{|t|+1}$.

In this paper, the predicted location on the road network is obtained by predicting both the next road segment ID *rid* (i.e., the unique identifier of roads) and the corresponding moving ratio *mr* from the start of the road segment, which denotes $\overrightarrow{p}_{|t|+1} = (rid_{|t|+1}, mr_{|t|+1})$. The moving ratio *mr* ranges from 0, indicating the vehicle is at the start of the specific road, to 1, indicating the vehicle is at the end. In this way, the map matching process can be avoided during model inference phase.

## 4. Method

### 4.1. Framework

We propose a flexible strategy of subgraph learning for predicting the next track point of vehicles. An overview of the proposed framework SLM is shown in Figure 2. There are three specific modules in the framework, which will be elaborated on in detail in this section.

### 4.2. Subgraph generation module

As described in the Introduction, precisely matching trajectory data to road networks is challenging. Therefore, using matched roads to represent trajectory points for trajectory prediction introduces the Uncertain Geographic Context Problem (UGCoP). To solve this problem, our model proposes using the surrounding roads of trajectory points, rather than matched roads, to represent trajectories. This approach helps to avoid the uncertainty associated with matched roads. In this section, neighboring roads around trajectory data are extracted as subgraphs by a subgraph generation module, and corresponding subgraph information is extracted for predicting the next track point. Specifically, given a road network $G = (V, E)$ and a trajectory point $p$, road segments in $G$ distributed within $\omega$ meters away from $p$ will be extracted as nodes of subgraph $G_p$ (see Figure 1f). Here, $\omega$ is a hyper-parameter that denotes the receptive field of trajectory points. Assume in total $s$ road segments are within $\omega$ meters away from $p$, denoted as $V_p = \{v_1, v_2, ..., v_s\}$, the connectivity relationship $E$ from $G$ will be extracted as edges of subgraph $G_p$, denoted as $E_p$.

Given the spatial structure of the subgraph $G_p = (V_p, E_p)$, its node features or representations also need to be defined. In previous studies, road embedding has proven to be an effective method for modeling intrinsic spatial relationships of road networks. Specifically, with a well-trained road network embedding table, each road segment will be represented by a unique $d$-dimensional vector, thus the similarities or relationships among roads can be accurately quantified. In this paper, for a road network $G = (V, E)$, we also create a road segment embedding table $D \in \mathbb{R}^{|V| \times d}$ to model it, and the corresponding embeddings of road segments in the subgraph $G_p$ can also be obtained, denoted as $D_p = \{D_{v_1}, D_{v_2}, ..., D_{v_s}\}$.

Finally, the information inherent in the subgraph $G_p = (V_p, E_p, D_p)$ is used to represent trajectory point $p$. According to the first law of geography, all geographic entities

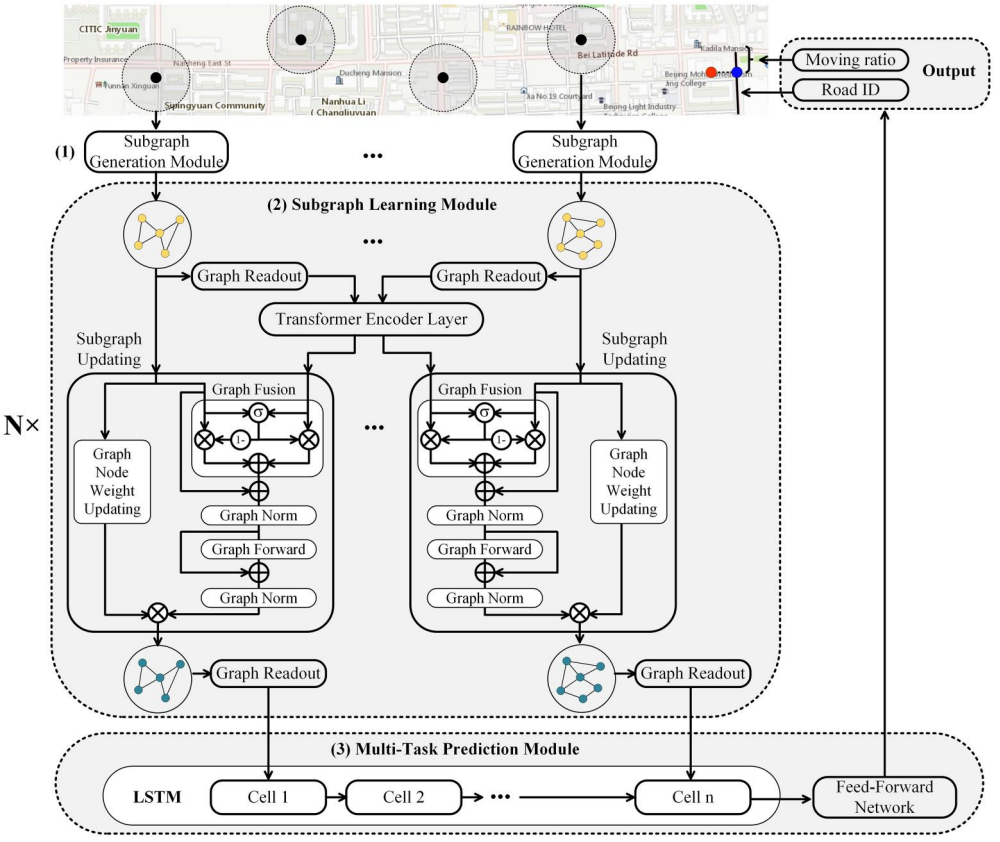


**Figure 2.** The framework of SLM includes three modules: (1) Subgraph generation module: given raw historical trajectory points of a vehicle, the subgraph generation module first extracts topology information of surrounding road networks. (2) Subgraph learning module: the features along with the subgraphs are passed through N subgraph learning modules to learn rich spatial and sequential features; in each module, a Transformer encoder layer is used to extract global sequential dependencies, and a subgraph updating process is designed for local spatial feature refinement. (3) Multi-task prediction module: given the output from the subgraph learning module, LSTM followed by a feed-forward network is designed to predict the target road ID and corresponding moving ratio.

are related, and the closer they are, the stronger the connection (Tobler 1970). Therefore, considering such a distance-decay effect, an exponential function is used to model the influence of each subgraph node on trajectory point $p$, and the representation of $p$ is defined in Equation 1, which is also the Graph Readout process in Figure 2.

$$g_p = \sum_{v_i \in V_p} w_{v_i} * D_{v_i} / \sum_{v_i \in V_p} w_{v_i}$$
$$w_{v_i} = \exp\left(-\text{dist}(v_i, p)/\varphi\right) \tag{1}$$

where $dist(v_i, p)$ denotes the distance between trajectory point $p$ and its projected point on the road segment $v_i$, and $\varphi$ is a hyper-parameter that controls the strength of the weight decay (i.e., $w_{v_i}$) as distance increases.

Therefore, given a raw historical trajectory $t = \{p_1, p_2, ..., p_{|t|}\}$, its initial representation can be defined as $g_t = \{g_1, g_2, ..., g_{|t|}\}$. To sum up, the output of the subgraph generation module is $(G_t, g_t) = \{(G_1, g_1), (G_2, g_2), ..., (G_{|t|}, g_{|t|})\}$.

### 4.3. Subgraph learning module

Given the output of the subgraph generation module, historical trajectory points are represented by corresponding subgraphs (i.e., ($G_t$, $g_t$)), thereby avoiding the uncertain problem of representing trajectories through map matching. Subsequently, we propose a subgraph learning module to learn rich spatial and sequential features of historical trajectory points. In this module, there are two parts including a Transformer encoder layer (Vaswani *et al*. 2017) and a subgraph updating layer. Specifically, the Transformer encoder layer is used to model sequential dependencies from the representation of historical trajectory points (i.e., $g_t$), and the subgraph updating layer is used for learning local spatial features of trajectories considering the connectivity of road segments in each subgraph (i.e., $G_t$).

#### 4.3.1. Transformer encoder layer

In the domain of NLP, the Transformer encoder layer has been widely employed to model text data, proving effective in analyzing and extracting sequential features (Gillioz *et al*. 2020, Devlin *et al*. 2018). Recently, some researchers have also introduced the Transformer encoder layer for trajectory analysis, considering that both trajectory data and text data are organized in a similar sequential manner (Giuliari *et al*. 2021, Chen *et al*. 2021b). These studies suggest that the Transformer encoder layer performs well in extracting sequential features from trajectories. Therefore, in this paper, given the representation $g_t = \{g_1, g_2, ..., g_{|t|}\}$ of a historical trajectory $t = \{p_1, p_2, ..., p_{|t|}\}$, the Transformer encoder layer (Vaswani *et al*. 2017) is used to model sequential dependencies.

$$\begin{aligned} z_t^{(l)} &= \text{TransformerEncoder}\left(h_t^{(l-1)}\right) \\ h_t^{(0)} &= g_t \end{aligned} \tag{2}$$

where $h_t^{(l-1)}$ denotes trajectory feature after $l$ – 1 layers of the subgraph learning module. In summary, the input to the Transformer encoder layer is the representation of all historical trajectory points (i.e., $g_t$), and the output is an updated representation of them (i.e., $z_t^{(l)}$), taking into account sequential dependencies. Details of the Transformer encoder layer are provided in the Appendix A for further reference.

#### 4.3.2. Subgraph updating layer

With the Transformer encoder layer, sequential dependencies among trajectory points can be effectively extracted. Raw trajectory points are generally offset with their real positions, which brings uncertain information in the initialization of subgraph features (i.e., $g_t$). In this section, a subgraph updating layer is designed for local spatial feature refinement. Given the output of the Transformer encoder layer $z_t^{(l)} = \left\{z_1^{(l)}, z_2^{(l)}, ..., z_{|t|}^{(l)}\right\}$ after $l$ layers of the subgraph learning module, local subgraph information will be

further refined based on the hidden feature $z_i^{(l)}$ of a trajectory point $p_i$ considering sequential dependencies.

Specifically, there are two objects including graph node embedding updating and graph node weight updating (Figure 2). Inspired by the Transformer encoder layer, three sub-layers including graph fusion, graph forward, and graph normalization are first designed for graph node embedding updating. Moreover, a sub-layer is specifically designed for updating the weights of graph nodes (initialized in Equation 1) to reduce the impact of uncertain measurement errors of trajectory positions.

***4.3.2.1. Graph fusion.*** In Section 4.3.1, the representations of all historical trajectory points (i.e., $g_t$) are input into the Transformer encoder layer. The updated representation of a trajectory point ($z_i^{(l)}$) contains global sequential information from its upstream and downstream trajectory points. In this section, a gated fusion mechanism (see Figure 2) is adopted to integrate the updated representation of each trajectory point and their corresponding local subgraph features (denoted as $f_i^{(l-1)}$). The subgraph of a trajectory point is constructed by the subgraph generation module, and local subgraph features are corresponding embedding information of graph nodes. In this way, the initial value of $f_i^{(0)}$ is $D_i$, which are obtained based on road network embedding defined in Section 4.2.

$$\begin{aligned} \rho_i^{(l)} &= \sigma\left(\ddot{z}_i^{(l)} W_z + f_i^{(l-1)} W_f + \tau\right) \\ Z_i^{(l)} &= \ddot{z}_i^{(l)} \rho_i^{(l)} + f_i^{(l-1)}\left(1 - \rho_i^{(l)}\right) \end{aligned} \tag{3}$$

where $\ddot{z}_i^{(l)}$ repeats $z_i^{(l)}$ for $|V_i|$ times to ensure that $\ddot{z}_i^{(l)}$ and $f_i^{(l-1)}$ share the same size, $W_z, W_f \in \mathbb{R}^{d\times d}$, $\tau$ are learnable parameters, and $\sigma()$ denotes the sigmoid activation function. In summary, the input of graph fusion includes the learned representation of trajectory point $p_i$ by the Transformer encoder layer ($z_i^{(l)}$) and local subgraph features ($f_i^{(l-1)}$), and the output is updated local subgraph features ($Z_i^{(l)}$) through the fusion of global and local features.

***4.3.2.2. Graph forward.*** After fusing global and local features (i.e., $Z_i^{(l)} \in \mathbb{R}^{|V_i|\times d}$), a graph forward sub-layer is designed for graph neighboring information propagation. In this layer, the GAT (Veličković *et al*. 2017) model is adopted. With this sub-layer, extracted topology information of neighboring roads (i.e., subgraph structure) can be used for graph node embedding updating. Graph Forward (*GFD*) can be described as follows:

$$\dot{f} = GFD(G, f) \tag{4}$$

where $G$ denotes a graph structure, $f$ denotes corresponding node features in this graph, and $\dot{f}$ denotes the updated node features considering spatial graph structure. Details of *GFD* are provided in the Appendix B for further reference.

***4.3.2.3. Graph norm.*** Subsequently, a graph normalization (*GN*) layer (Dwivedi and Bresson 2020), specifically designed for graphs, is applied based on the output of graph fusion or graph forward to enhance the stability and convergence of model training. The process can be described as follows:

$$\begin{aligned} \dot{f}_i^{(l)} &= GN\left(f_i^{(l-1)} + Z_i^{(l)}\right) \\ f_i^{(l)} &= GN\left(\dot{f}_i^{(l)} + GFD(G_i, \dot{f}_i^{(l)})\right) \end{aligned} \tag{5}$$

where $Z_i^{(l)}$ is the output of graph fusion, $G_i$ denotes the spatial structure of generated subgraph of trajectory point $i$, $\dot{f}_i^{(l)}$ and $f_i^{(l)}$ denote the output of the first and second graph norm layer in Figure 2, respectively. Details of the graph norm layer are provided in the Appendix C for further reference.

***4.3.2.4. Graph node weight updating.*** Graph node weight updating is also proposed in this module. In the subgraph generation module, initial graph node weight information is obtained based on distances between a raw trajectory point and neighboring road segments (Equation 1). However, with measurement errors, raw trajectory points are generally offset with their real positions, which brings errors when calculating graph node weight. Moreover, though spatial objects with direct connections are related, there are usually limited neighbors that have strong correlations (Yu *et al*. 2022). Therefore, graph node weight updating is designed to locate neighbors with strong correlations and reduce the effect of noise and less related roads. In this paper, a graph forward layer with a sigmoid function is used to update graph node weight.

$$w_i^{(l)} = \sigma\left(GFD(G_i, h_i^{(l-1)})\right) \tag{6}$$

where $G_i$ denotes the spatial structure of generated subgraph of trajectory point $i$, *GFD* and $\sigma$ denote Graph forward and sigmoid function, respectively. It should be noted that the initial value of $w_i^{(0)}$ is obtained based on Equation 1.

## 4.4. Multi-task prediction module

After extracting rich spatial and sequential features of the historical raw trajectory of a vehicle, a multi-task prediction module is proposed to predict its next location on a road. Specifically, there are two tasks in this module, including predicting the next road and the corresponding moving ratio on this road. The moving ratio *mr* ranges from 0, indicating the vehicle is at the start of the specific road, to 1, indicating the vehicle is at the end. In this way, the next track point on road networks can be obtained directly, without any postprocessing steps such as map matching, which can also avoid associated uncertainty. In this module, given the output from the subgraph learning module, the LSTM (Hochreiter and Schmidhuber 1997) will be further utilized to model the trajectory sequence. Specifically, the input to this module is the updated trajectory representation after subgraph learning, This process is defined as follows:

$$\begin{aligned} h_i^{(N)} &= w_i^{(N)} f_i^{(N)} \\ \Gamma &= LSTM(h_1^{(N)}, h_2^{(N)}, \ldots, h_{|t|}^{(N)}) \end{aligned} \tag{7}$$

where $N$ is the number of subgraph learning modules, and details of the LSTM are provided in the Appendix D for further reference.

After that, two feed-forward layers are used for predicting the probability distribution indicating the likelihood of the next track point being located on each road ($\widehat{P_{rid}}$),

as well as the corresponding moving ratio ($\widehat{mr}$), respectively.

$$\begin{aligned} \widehat{P_{rid}} &= \Gamma W_r + \tau_r \\ \widehat{mr} &= \sigma(\Gamma W_m + \tau_m) \end{aligned} \tag{8}$$

where $W_r \in \mathbb{R}^{d\times|V|}$, $|V|$ denotes the number of roads in the focused road network, $W_m \in \mathbb{R}^{d\times 1}$, $\tau_r$, and $\tau_m$ are learnable parameters. Here, $\sigma()$ denotes the sigmoid activation function, ensuring the value of $mr$ ranges from 0 to 1. The predicted road segment ID is the one with the maximum probability value in $P_{rid}$ (i.e., $rid = argmax(P_{rid})$).

Finally, we adopt the cross entropy $L_{ce}$ and the mean square loss $L_{mse}$ as the loss functions for road segment ID and moving ratio prediction tasks, respectively. The final loss function can be described as follows.

$$\begin{aligned} Loss &= L_{ce}(\widehat{P_{rid}}, P_{rid}) + \lambda L_{mse}(\widehat{mr}, mr) \\ L_{ce}(\hat{y}_i\ y_i) &= -\frac{1}{n}\sum_{i}^{n}\left[(1-y_i)\log(1-\hat{y}_i) + y_i \log \hat{y}_i\right] \\ L_{mse}(\hat{y}_i\ y_i) &= \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2 \end{aligned} \tag{9}$$

where $P_{rid}$ and $mr$ are corresponding ground truth, $\lambda$ is a hyper-parameter.\o-penup 2pt

### 4.5. Workflow of model training

Finally, to clearly present the workflow of model training, a pseudocode framework is provided in Algorithm 1. During model training, the input data consists of a road network $G = (V, E, D)$, raw historical trajectory points of a vehicle $t = \{p_1, p_2, ..., p_{|t|}\}$, and the ground truth of the next track point $\overrightarrow{p}_{|t|+1} = (rid_{|t|+1}, mr_{|t|+1})$. The input data flow starts from the subgraph generation module to the subgraph learning module, and finally to the multi-task prediction module. The obtained loss is used to optimize model parameters by comparing the output prediction result with the ground truth.

**Algorithm** 1: The workflow for training SLM

**Data:** A road network $G = (V, E, D)$; raw historical trajectory points of a vehicle $t = \{p_1, p_2, ..., p_{|t|}\}$; the ground truth of the next track point $\overrightarrow{p}_{|t|+1} = (rid_{|t|+1}, mr_{|t|+1})$;

$G_t, g_t$ = Subgraph generation module(G, t); // see Equation (1)
**for** $l \leftarrow 1$ ***to*** $N$ **do**
  // Subgraph learning module
  $z_t^{(l)}$ = TransformerEncoder $(h_t^{(l-1)})$; // $h_t^{(0)} = g_t$
  **for** $i \leftarrow 1$ ***to*** $|t|$ **do**
    // Subgraph updating module
    $Z_i^{(l)}$ = Graph Fusion$(z_i^{(l)}, f_i^{(l-1)})$; // $f_i^{(0)} = D_i$
    $\dot{f}_i^{(l)}$ = Graph Norm$(f_i^{(l-1)} + Z_i^{(l)})$;
    $f_i^{(l)}$ = Graph Norm$(\dot{f}_i^{(l)}$ + Graph Forward$(G_i, \dot{f}_i^{(l)}))$;
    $w_i^{(l)}$ = Graph Node Weight Updating$(G_i, h_i^{(l-1)})$;
    $h_i^{(l)} = w_i^{(l)} f_i^{(l)}$;
$\widehat{P_{rid_{|t|+1}}}, \widehat{mr_{|t|+1}}$ = Multi-task prediction module$(h_1^{(N)}, h_2^{(N)}, ..., h_{|t|}^{(N)})$;

// $\widehat{P_{rid_{|t|+1}}}$ represents the probability distribution indicating the likelihood of the next track point being located on each road
// $\widehat{rid}_{|t|+1} = argmax(\widehat{P_{rid_{|t|+1}}})$
**11** $Loss = L_{ce}(\widehat{P_{rid_{|t|+1}}}, P_{rid_{|t|+1}}) + \lambda L_{mse}(\widehat{mr}_{|t|+1}, mr_{|t|+1})$;

## 5. Experiment

### 5.1. Experiment setting

#### 5.1.1. Datasets

Our experiments are based on road networks and trajectory datasets from two cities, namely Beijing and Porto. The road networks are obtained from OpenStreetMap (https://www.openstreetmap.org/). The trajectory datasets are recorded on taxis in these two cities. In practice, both the input and output of our model do not require map matching during model inference (see Section 4). However, during model training, the accurate position of the next track point on road segments needs to be specified to calculate the loss used for optimizing model parameters (see Equation 9). To obtain training data, a widely used map matching method (Yang and Gidofalvi 2018) is adopted to transform trajectory points into sequences of positions on road segments (i.e., road segment IDs and moving ratios). Only the matching results of the next track points are used as ground truth to train and evaluate our method. However, as discussed in the preceding analysis, precisely matching trajectories to roads is a challenge in ITS, as the matching precision is greatly affected by the quality of the trajectory. Therefore, we aim to obtain high-quality trajectories by preprocessing the datasets and filtering out trajectories with noticeable noises (e.g., track points with over fast and over low car speed, excessive distance from a last track point to a next track point). Additionally, trajectories with less than 10 minutes in both trajectory datasets are filtered out to avoid uncertainty in short-time trajectories. Finally, the matching results are also manually checked. The time interval is set as 1 minute and statistics of the datasets are shown in Table 1.

Each dataset is partitioned along the time axis into three non-overlapping parts, including training dataset, validation dataset, and testing dataset, with a ratio of 2:1:1. After filtering out relatively low-quality trajectories in both datasets, to simulate trajectories in a realistic environment, Gaussian noise with a radius of 30 meters is added to a random half of the trajectories in both the training and validation datasets. (Li *et al.* 2018).

$$\begin{aligned} \check{p_x} &= p_x \pm 30 \times d_x, \qquad d_x \sim Gaussian(0,1) \\ \check{p_y} &= p_y \pm 30 \times d_y, \qquad d_y \sim Gaussian(0,1) \end{aligned} \tag{10}$$

Similar Gaussian noise is added to all testing datasets to obtain testing datasets with noise. In this way, the performance of our method on testing datasets with or

**Table 1.** Statistics of datasets.

| Dataset | Beijing | Porto |
|---|---|---|
| #Trajectories | 102,375 | 151,041 |
| #Road Segments | 36,189 | 36,898 |
| #Trajectory collected time | June 2013 | July 2013-June 2014 |

without noise will be evaluated. We also adopt map matching on datasets with noise to obtain corresponding location sequences on road segments.

### 5.1.2. Evaluation metrics

The task we focus on is to accurately predict the next track point on road segments of a vehicle given historical raw trajectory points. Therefore, three evaluation metrics are adopted. Firstly, the Euclidean distance (also known as L2 distance) between the predicted location and the true location is used to evaluate the performance of different methods. Secondly, since urban regions are connected by road networks, it may take a considerable amount of time to travel from one location to another, even if they are closely distributed in Euclidean space (Yu *et al*. 2019; Zhang *et al*. 2022). Therefore, because vehicles navigate through road networks, the road network distance is also used for evaluation. Moreover, the accuracy of predicting road segment ID is used as an evaluation metric.

$$\begin{aligned} L2\_dis &= \frac{\sum_{i=1}^{lp} L2\_dist\left(\widehat{\overrightarrow{p}}_i, \overrightarrow{p}_i\right)}{l_p} \\ rn\_dis &= \frac{\sum_{i=1}^{lp} rn\_dist\left(\widehat{\overrightarrow{p}}_i, \overrightarrow{p}_i\right)}{l_p} \\ acc &= \frac{\sum_{i=1}^{l_p} 1(\widehat{rid}_i = rid_i)}{l_p} \end{aligned} \tag{11}$$

where $\overrightarrow{p}_i$ is the true location on a road, $\widehat{\overrightarrow{p}}_i$ is the predicted location, $l_p$ is the size of locations. $L2\_dist()$ is a function to calculate the Euclidean distance between the true location and predicted location, and $rn\_dist()$ is a function to calculate the road network distance between them. Specifically, given two points on the road network, $rn\_dist()$ is obtained by calculating their shortest path distance using Dijkstra's algorithm (Dijkstra 1959).

### 5.1.3. Baselines

In our experiments, we compare our method with recent representative methods, including road representation methods and trajectory representation methods.

**DW** (Perozzi *et al*. 2014): DW learns road representations by first transforming the road network into node sequences by random walk, and subsequently applying the skip-gram model on node sequences. The walk length, the number of walks per node, and the window size are set to 30, 25, and 5 respectively.

**node2vec** (Grover and Leskovec 2016): node2vec employs a biased random walk strategy to explore the neighborhood of a node, and also applies the skip-gram model on the node sequences. The walk length, the number of walks per node, and the window size are set to be the same as in DW. The biased random walk parameters $p$ and $q$ in this method are tuned in the set of $\{1/2, 1, 2\}$.

**GAE** (Kipf and Welling 2016): GAE uses a graph convolutional network (GCN) to learn node representations with an autoencoder structure, which is trained to reconstruct the original graph structure. The model consists of two GCN layers, and the

dimension of the output hidden vectors of the first layer is set as 256. In this method, road segments and corresponding connections are regarded as graph nodes and edges, respectively.

**t2vec** (Li *et al*. 2018): t2vec is a trajectory representation method, which is trained with an encoder-decoder framework. In the training phase, the cell size is set as 100 meters, and original parameter settings in Li *et al*. (2018) are adopted. We finally obtain 28,028 cells in the Beijing dataset and 15,743 cells in the Porto dataset. After the model is well-trained, we use only the encoder for trajectory representation.

**T3S** (Yang *et al*. 2021): T3S is a trajectory representation method that takes both trajectory points and cell indexes as input. We adopt the same grid information with t2vec, and original parameter settings in Yang *et al*. (2021) are also adopted.

**Toast (**Chen et al. 2021b**)**: Toast first uses context-aware node2vec to learn road embedding, and subsequently trains a Transformer encoder for trajectory representation. Original parameter settings in [20] are adopted.

As for DW, node2vec, and GAE, we first learn road embeddings based on these methods; subsequently, matched road segments based on historical trajectory points will be regarded as input; finally, our designed multi-task prediction module will be used to predict the next locations. As for T3S, the feed-forward networks in the multi-task prediction module are used for predicting the next location based on obtained hidden vectors by T3S. As for t2vec and Toast, these two models are firstly pre-trained, and subsequently used for location prediction with the feed-forward networks in the multi-task prediction module. It should be noted that both road embeddings and well-trained trajectory representation models will be further fine-tuned during the location prediction training phase for a fair comparison.

#### *5.1.4. Parameter settings*

In our experiments, both our method and baselines are implemented in the Python Pytorch framework, and four former trajectory points are used as historical points to predict the next track point. We set the size of all hidden-state vectors d to 128 for all methods for a valid comparison. As for our method, the number of subgraph learning module N is set as 2. The hyper-parameters $\omega$ and $\varphi$ are set as 200 and 100 meters, respectively, the number of attention heads in both the Transformer encoder layer and graph attention layer is set as 8, and $\lambda$ is set as 10. Both our method and baselines are trained with Adam optimizer for 60 epochs with batch size 32 and learning rate $10^{-4}$. All the experiments are conducted on a machine with Intell Core i9-10900X CPU and NVIDIA GeForce RTX 2080Ti GPU.

### *5.2. Performance evaluation*

In this section, our proposed SLM is compared against baselines, and the results are listed in Table 2. The best results of each evaluation index achieved among all models are marked in bold. To further show the performance of each model when dealing with datasets with noise, the difference value between evaluation indexes obtained based on datasets with and without noise are calculated as $\delta_{acc} = acc - acc(noise)$, $\delta_{rn\_dis} = rn\_dis(noise) - rn\_dis$, and $\delta_{L2\_dis} = L2\_dis(noise) - L2\_dis$. We have

**Table 2.** Performance comparison of SLM and baseline models on the Beijing dataset (BJ) and Porto dataset (PT). The units of rn_dis and L2_dis are meters, acc (noise), rn_dis (noise), and L2_dis (noise) are obtained based on datasets with additional noise by Equation 10, $\delta_{\text{acc}}$ is obtained by acc - acc (noise), $\delta_{\text{rn\_dis}}$ is obtained by rn_dis(noise)-rn_dis, and $\delta_{\text{L2\_dis}}$ is obtained by L2_dis(noise)-L2_dis.

| Data | Method | acc | acc (noise) | $\delta_{\text{acc}}$ | rn_dis | rn_dis (noise) | $\delta_{\text{rn\_dis}}$ | L2_dis | L2_dis (noise) | $\delta_{\text{L2\_dis}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| BJ | deepwalk | 0.410 | 0.388 | 0.022 | 978.6 | 1015.7 | 37.0 | 304.5 | 310.8 | 6.3 |
| | node2vec | 0.407 | 0.386 | 0.021 | 987.9 | 1021.4 | 33.5 | 308.1 | 313.0 | 4.9 |
| | GAE | 0.396 | 0.372 | 0.024 | 1029.5 | 1074.2 | 44.8 | 377.7 | 393.2 | 15.6 |
| | t2vec | 0.397 | 0.382 | 0.015 | 1063.1 | 1106.6 | 43.6 | 534.8 | 589.2 | 54.4 |
| | T3S | 0.317 | 0.296 | 0.021 | 1284.8 | 1363.9 | 79.1 | 794.9 | 907.6 | 112.7 |
| | toast | 0.386 | 0.363 | 0.023 | 1092.9 | 1139.2 | 46.4 | 524.2 | 565.0 | 40.7 |
| | SLM | 0.436 | 0.424 | 0.012 | 634.3 | 658.6 | 24.4 | 283.9 | 291.9 | 8.0 |
| Data | Method | acc | acc (noise) | $\delta_{\text{acc}}$ | rn_dis | rn_dis (noise) | $\delta_{\text{rn\_dis}}$ | L2_dis | L2_dis (noise) | $\delta_{\text{L2\_dis}}$ |
| PT | deepwalk | 0.3013 | 0.279 | 0.022 | 932.5 | 966.0 | 33.6 | 288.1 | 289.8 | 1.8 |
| | node2vec | 0.3005 | 0.279 | 0.022 | 933.1 | 966.6 | 33.5 | 283.6 | 285.7 | 2.0 |
| | GAE | 0.294 | 0.270 | 0.023 | 970.4 | 1005.0 | 34.6 | 345.9 | 349.5 | 3.6 |
| | t2vec | 0.278 | 0.267 | 0.0101 | 1000.3 | 1022.6 | 22.3 | 367.0 | 381.6 | 14.5 |
| | T3S | 0.237 | 0.225 | 0.011 | 1094.5 | 1124.7 | 30.2 | 442.8 | 467.2 | 24.3 |
| | toast | 0.291 | 0.269 | 0.023 | 975.0 | 1012.3 | 37.4 | 336.3 | 344.2 | 7.8 |
| | SLM | 0.296 | 0.286 | 0.0097 | 391.7 | 404.0 | 12.3 | 262.6 | 269.6 | 6.9 |

the following observations: (1) Our method can achieve the best performance on most evaluation indexes on these datasets. Though our method obtains a slightly worse *acc* result than deepwalk and node2vec on the Porto dataset, our method can still obtain much better results in terms of *rn_dis* and *L2_dis* than all the baselines on this dataset. (2) Our method achieves better results on most of the difference values between datasets with and without noise (i.e., $\delta_{acc}$ and $\delta_{rn\_dis}$), and competitive but slightly higher values in terms of $\delta_{L2\_dis}$. This demonstrates the relative robustness of our method to noise in trajectories, which also validates the effectiveness of our proposed strategy of subgraph learning. (3) Methods based on map matching (deepwalk, node2vec, GAE, and toast) obtain similar $\delta_{acc}$ values, and the results are worse than t2vec, which encodes trajectories based on spatial grid. Therefore, compared with road segment, using spatial grid as the carrier to represent trajectories can be more robust to noise when predicting which road segment the next location will be on. However, such a conclusion seems not to be suited for $\delta_{rn\_dis}$ results, and we guess this phenomenon may be caused by the overlook of road network structure in t2vec. However, by using the subgraph rather than grid or solely matched road segments to represent trajectories, our method can be more robust to both evaluation indexes.

### 5.3. Ablation study

To prove the effectiveness of our proposed modules, five variants of SLM are compared. **w/o SU** replaces the Subgraph Updating modules with the transformer encoder layer, which means the subgraph structure will not be used now. **w/o GF** replaces the Graph Fusion module with a concatenation operation and a feed-forward network. **w/o GFD** removes Graph ForwarD in the graph node embedding updating module. **w/o GN** removes Graph Norm. **w/o GNWU** removes the Graph Node Weight Updating

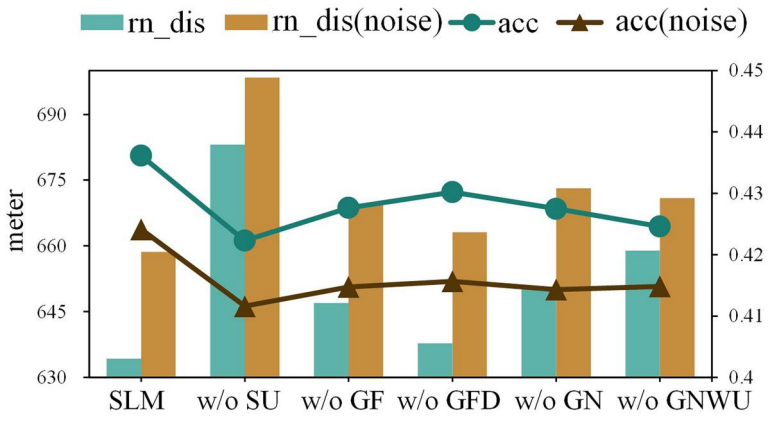


**Figure 3.** Results of ablation studies.

module. The results are listed in Figure 3, and it can be observed that SLM consistently outperforms all its variants, which proves the significance of these modules.

## 5.4. Parameter analysis

In this section, the impact of hyper-parameters including the number of subgraph learning module (N), the receptive field of trajectory points ($\omega$), and the parameter that controls the strength of the weight decay as distance increases ($\varphi$ in Equation 1). The results are listed in Figure 4.

### 5.4.1. The impact of the number of subgraph learning module (N)

The number of subgraph learning module (N) directly influences the complexity of SLM. To examine its impact, we vary N in SLM from 1 to 4 and report the results in Figures 4a and d. Based on the results, our SLM shows consistent performance across various values of N on both the Beijing and Porto datasets, while the performance of *rn_dis* and *rn_dis*(*noise*) shows a slight improvement with increasing N on the Porto dataset. Therefore, considering both effectiveness and efficiency, $N = 2$ is chosen in our experiment.

### 5.4.2. The influence of the receptive field of trajectory points ($\omega$)

The receptive field of trajectory points determines how much neighboring roads will be considered during the following subgraph learning. In practice, too few roads cannot reveal implicit urban context while too many roads will bring chaos. From the results in Figures 4b and e, setting the receptive field as 200 meters yields the best results on the Beijing dataset, while setting the receptive field as 400 meters produces the best results on the Porto dataset. To further investigate the potential reasons for this result, we calculate the road network densities used in our experiment for these two datasets. Specifically, the road network density of the Beijing dataset is 0.016 $m/m^2$, while the density of the Porto dataset is 0.011 $m/m^2$. Consequently, setting the receptive field as 200 meters may capture enough neighboring road segments in the Beijing dataset, while obtaining enough road segments requires a larger receptive

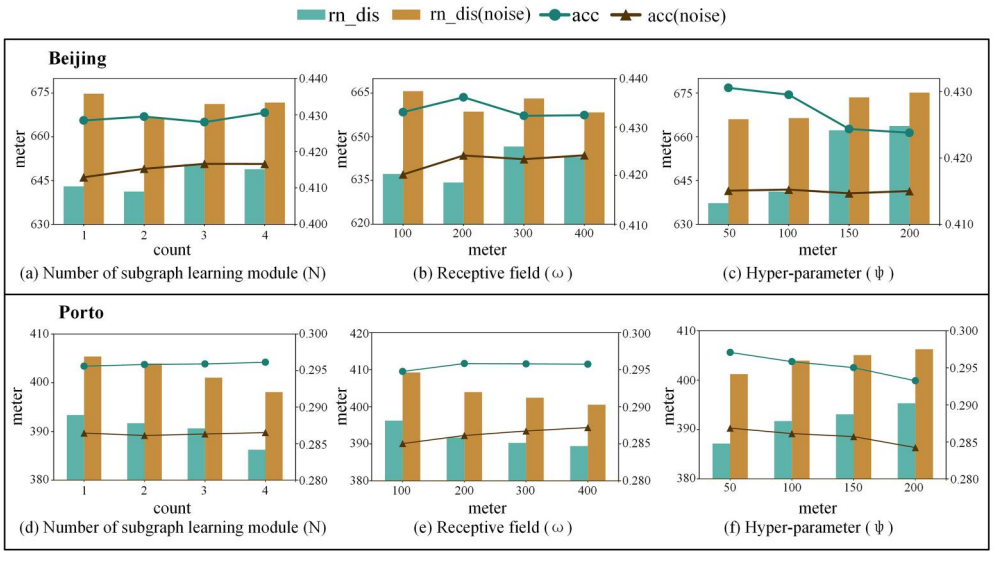


**Figure 4.** The performance of SLM on Beijing and Porto datasets under different hyper-parameters.

field in the Porto dataset (e.g., 400 meters). Therefore, selecting an appropriate value for the receptive field enables the subgraphs, composed of neighboring road segments, to better represent corresponding trajectory points. Moreover, despite obtaining different performances with different receptive fields, our method consistently outperforms the baselines, validating our subgraph learning strategy.

#### *5.4.3. The influence of the hyper-parameter (φ)*

When it comes to the hyper-parameter $\varphi$, assigning a small value to it will result in large weights being given to nearby roads and small weights being given to distant roads; while assigning a large value to $\varphi$ will result in even larger weights being given to nearby roads, while relatively larger weights will also be assigned to distant roads compared to what is obtained when a small $\varphi$ is used. Therefore, from the results in Figures 4c and f, when there exists no noise in datasets, a small value of $\varphi$ (e.g., 50 meters) can force the model to focus on near roads, which may be the potential travel route of corresponding trajectory points. However, when there are offsets in trajectory points, the model will be unstable by focusing mainly only near roads. Therefore, to avoid measurement errors, we set $\varphi$ as 100 meters in our experiments.

## 6. Discussion

### *6.1. Limitations*

While our method shows promise, it is crucial to acknowledge its limitations. Firstly, to validate our idea, we utilized a widely-used map matching method to project vehicles onto road networks and obtained the training and evaluation datasets. However, as discussed in the Introduction, the precision of map matching is significantly influenced by the quality of trajectory data, leading to uncertain matching errors in the obtained

labels, which is a limitation of this study. In our experiments, we attempted to mitigate this limitation by preprocessing the datasets and filtering out trajectories with noticeable noise. This allowed us to achieve relatively high-quality trajectory matching results based on the processed datasets. Moreover, the matching results were also manually checked. Consequently, the input trajectory data, along with their labels (the accurate trajectory points on the road network), could be used to train our prediction model. However, since trajectory data with noticeable noise were filtered out, the remaining training trajectory data are of relatively high quality, which may not fully reflect the trajectory data encountered in real-world scenarios. To simulate noise, our experiments introduced Gaussian noise to the trajectories. However, it is worth noting that Gaussian noise might not entirely represent the full spectrum of real-world noise types. In future research, we plan to consider exploring other noise models to evaluate the robustness of our findings. Moreover, though the obtained labels are of high quality, some residual influence on the results still remains due to the inevitably remaining noises. To further enhance the reliability of our experiments and conclusions, it is imperative to gather more accurate label datasets for training our model in future work. One possible approach is to associate trajectory points with corresponding photographs captured by dashcams on cars. This association could allow the accurate positions of vehicles on road networks to be inferred, serving as true labels to train and evaluate our model (Chang *et al*. 2016).

### *6.2. Broader application*

In this paper, we propose using subgraphs constructed by neighboring roads around trajectories for trajectory representation, and design three modules based on the subgraph structure for predicting the next track point. Our strategy takes into account the constraints of road networks and avoids the uncertainty introduced by matching trajectories onto roads. While our primary focus is on predicting the next track point, our strategy holds promise for development in other tasks where geographical context is significant. For instance, trajectory similarity computation and travel time prediction are key challenges in ITS (Fu and Lee 2020, Li *et al*. 2018, Yang *et al*. 2021, Han *et al*. 2021), and both tasks are heavily influenced by geographical context. Therefore, our method has the potential to be transferred to these tasks. However, since our model requires training data to adapt to these tasks, a key aspect for broader application is the collection of reliable training data and corresponding labels from the real world. As discussed in Section 6.1, we intend to enhance the robustness of our training data by integrating additional data types, such as photographs captured by dashcams on cars. Therefore, it is essential to devise efficient strategies for collecting training data for corresponding tasks to facilitate broader application of our method.

## 7. Conclusion

In this paper, we propose a flexible strategy of subgraph learning called SLM for the next track point prediction of vehicles on road networks. Three modules are designed in SLM, including a subgraph generation module for extracting topology information of neighboring roads around trajectories, a subgraph learning module for learning

global sequential features and local spatial features, and a multi-task prediction module for predicting the next track point of vehicles on road networks. With the SLM, the uncertain contextual information of neighboring roads around historical trajectory points can be used, and the exact location of vehicles on road networks can be directly obtained by SLM without any pre- or post-processing (e.g., map matching) during model inference phase. Moreover, since the challenge of map matching can be avoided, our method presents to be more robust to trajectory datasets with noise. Experiments based on trajectory datasets from two cities validate our conclusion.

## Acknowledgement(s)

The authors are grateful to the associate editor, Bo Huang, and the anonymous referees for their valuable comments and suggestions.

## Authors' contributions

Yifan Zhang: Methodology, validation, formal analysis, investigation, writing-original draft preparation, writing-review and editing, visualization; Wenhao Yu: Conceptualization, formal analysis, writing-original draft preparation, writing-review and editing, supervision, project administration, funding acquisition; Dizhu: Writing-review and editing, formal analysis, supervision. All authors have read and agreed to the published version of the manuscript.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

The project was supported by the National Natural Science Foundation of China (42371446 and 42071442), Natural Science Foundation of Hubei Province (2024AFD412), and by the Fundamental Research Funds for the Central Universities, China University of Geosciences (Wuhan) (No. CUG170640). This research was also supported by Meituan. Di Zhu was supported by DSI Medium/Large Seed Grant 2023, Data Science Initiatives, University of Minnesota (1701-10964-20090-5672018-UMF0024430).

## Notes on contributors

***Yifan Zhang*** is a Ph.D. student in the School of Geography and Information Engineering, China University of Geosciences, Wuhan, China (CUG). His research interests include deep learning and spatial data mining.

***Wenhao Yu*** received the B.S. and Ph.D. degrees in Geoinformatics from the Wuhan University, Wuhan, China, in 2010 and 2015, respectively. He is a professor at China University of Geosciences, Wuhan, China (CUG). His research interests include spatial data mining, map generalization, and deep learning.

***Di Zhu*** is an Assistant Professor of Geographic Information Science at the University of Minnesota, Twin Cities (UMN). His research aims at generating both theoretical and actionable

insights from spatiotemporal data by exploring the frontiers that bridge geospatial analysis, artificial intelligence and social sensing.

## ORCID

Yifan Zhang http://orcid.org/0000-0002-5328-0881
Di Zhu http://orcid.org/0000-0002-3237-6032

## Data and codes availability statement

The data and codes that support the findings of this study are available with a DOI at (https://doi.org/10.6084/m9.figshare.22785503).

## References

Altch, F., and Fortelle, A., 2017. An lstm network for highway trajectory prediction. *In*: *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 353–359.

Ba, J.L., Kiros, J.R., and Hinton, G.E., 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bacciu, D., *et al.*, 2023. Safety and robustness for deep neural networks: An automotive use case. *In*: J. Guiochet, S. Tonetta, E. Schoitsch, M. Roy and F. Bitsch, eds. *Computer Safety, Reliability, and Security. SAFECOMP 2023 Workshops*, Cham: Springer Nature Switzerland, 95–107.

Bacciu, D., *et al.*, 2017. An experience in using machine learning for short-term predictions in smart transportation systems. *Journal of Logical and Algebraic Methods in Programming*, 87, 52–66. Available from: https://www.sciencedirect.com/science/article/pii/S2352220816301584.

Chang, S.P., *et al.*, 2016. Extracting driving behavior: Global metric localization from dashcam videos in the wild. *In*: G. Hua and H. J'egou, eds. *Computer vision – ECCV 2016 workshops*. Cham: Springer International Publishing, 136–148.

Chekol, A.G., and Fufa, M.S., 2022. A survey on next location prediction techniques, applications, and challenges. *EURASIP Journal on Wireless Communications and Networking*, 2022 (1), 29.

Chen, M., *et al.*, 2021a. Origin-aware location prediction based on historical vehicle trajectories. *ACM Transactions on Intelligent Systems and Technology*, 13 (1), 1–18.

Chen, M., *et al.*, 2021b. Cem: A convolutional embedding model for predicting next locations. *IEEE Transactions on Intelligent Transportation Systems*, 22 (6), 3349–3358.

Chen, M., Yu, X., and Liu, Y., 2019a. Mpe: a mobility pattern embedding model for predicting next locations. *World Wide Web*, 22 (6), 2901–2920.

Chen, P., *et al.*, 2019b. Stlp-gsm: a method to predict future locations of individuals based on geotagged social media data. *International Journal of Geographical Information Science*, 33 (12), 2337–2362.

Chen, X., and Kwan, M.P., 2015. Contextual uncertainties, human mobility, and perceived food environment: The uncertain geographic context problem in food access research. *American Journal of Public Health*, 105 (9), 1734–1737.

Chen, Y., *et al.*, 2021b. Robust road network representation learning: When traffic patterns meet traveling semantics. *In*: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21*. New York, NY: Association for Computing Machinery, 211–220. https://doi.org/10.1145/3459637.3482293.

Chen, Y., *et al.*, 2022. Rntrajrec: Road network enhanced trajectory recovery with spatial-temporal transformer. *arXiv preprint arXiv:2211.13234*.

Chen, Z., *et al.*, 2019c. A novel sparse representation model for pedestrian abnormal trajectory understanding. *Expert Systems with Applications*, 138, 112753.

Cui, G., Luo, J., and Wang, X., 2018. Personalized travel route recommendation using collaborative filtering based on gps trajectories. *International Journal of Digital Earth*, 11 (3), 284–307.

Dai, J., *et al.*, 2015. Personalized route recommendation using big trajectory data. *In*: *IEEE 31st international conference on data engineering,* 13–17 April 2015 Seoul, Korea. IEEE, 543–554.
De Caro, V., *et al.*, 2023. Prediction of driver's stress affection in simulated autonomous driving scenarios. *In*: *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, Rhodes Island, Greece, 1–5.
Demšar, U., and Virrantaus, K., 2010. Space–time density of trajectories: exploring spatio-temporal patterns in movement data. *International Journal of Geographical Information Science*, 24 (10), 1527–1542.
Devlin, J., *et al.*, 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 (1), 269–271. https://doi.org/10.1007/BF01386390.
Dwivedi, V.P., and Bresson, X., 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.
Ebel, P., *et al.*, 2020. Destination prediction based on partial trajectory data. *In*: *2020 IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV, USA. IEEE, 1149–1155.
Fu, T.Y., and Lee, W.C., 2020. Trembr: Exploring road networks for trajectory representation learning. *ACM Transactions on Intelligent Systems and Technology*, 11 (1), 1–25.
Gillioz, A., *et al.*, 2020. Overview of the transformer-based models for nlp tasks. *In: 2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, Sofia, Bulgaria. IEEE, 179–183.
Giuliari, F., *et al.*, 2021. Transformer networks for trajectory forecasting. *In*: *2020 25th international conference on pattern recognition (ICPR),* Milan, Italy. IEEE, 10335–10342.
Grover, A., and Leskovec, J., 2016. node2vec: Scalable feature learning for networks. *In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY: Association for Computing Machinery, 855–864. https://doi.org/10.1145/2939672.2939754.
Guo, S., *et al.*, 2018. Trajectory prediction for ocean vessels base on k-order multivariate markov chain. *In: Wireless Algorithms, Systems, and Applications: 13th International Conference, WASA 2018, Tianjin, China,* June 20-22, *2018, Proceedings 13*. Springer, 140–150.
Han, P., *et al.*, 2021. A graph-based approach for trajectory similarity computation in spatial networks. *In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. New York, NY: Association for Computing Machinery, 556–564. https://doi.org/10.1145/3447548.3467337.
Hochreiter, S., and Schmidhuber, J., 1997. Long short-term memory. *Neural Computation*, 9 (8), 1735–1780.
Huang, Y., *et al.*, 2022. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7 (3), 652–674.
Huang, Z., *et al.*, 2021. Survey on vehicle map matching techniques. *CAAI Transactions on Intelligence Technology*, 6 (1), 55–71.
Huang, Z., *et al.*, 2023. Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving. *IEEE Transactions on Neural Networks and Learning Systems*, PP, 1–14.
Jiang, J., *et al.*, 2022. Self-supervised trajectory representation learning with temporal regularities and travel semantics. *arXiv preprint arXiv:2211.09510*.
Kipf, T.N., and Welling, M., 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.
Kong, D., and Wu, F., 2018. Hst-lstm: A hierarchical spatial-temporal long-short term memory network for location prediction. *In: IJCAI*, 18, 2341–2347.
Kontopoulos, I., Makris, A., and Tserpes, K., 2023. Traclets: A trajectory representation and classification library. *SoftwareX*, 21, 101306.
Kwan, M.P., 2012a. How gis can help address the uncertain geographic context problem in social science research. *Annals of GIS*, 18 (4), 245–255.
Kwan, M.P., 2012b. The uncertain geographic context problem. *Annals of the Association of American Geographers*, 102 (5), 958–968.

Li, B., *et al.*, 2021. A trajectory restoration algorithm for low-sampling-rate floating car data and complex urban road networks. *International Journal of Geographical Information Science*, 35 (4), 717–740.

Li, B., *et al.*, 2023. Vis-mm: a novel map-matching algorithm with semantic fusion from vehicle-borne images. *International Journal of Geographical Information Science*, 37 (5), 1069–1098.

Li, X., *et al.*, 2018. Deep representation learning for trajectory similarity computation. *In*: *2018 IEEE 34th international conference on data engineering (ICDE)*, Paris, France. IEEE, 617–628. https://doi.org/10.1109/ICDE.2018.00062.

Liang, R., *et al.*, 2023. Stglow: A flow-based generative framework with dual-graphormer for pedestrian trajectory prediction. *IEEE Transactions on Neural Networks and Learning Systems*, PP, 1–14.

Lin, Y., *et al.*, 2021. Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35 (5), 4241–4248.

Lin, Y., *et al.*, 2022. Contrastive pre-training of spatial-temporal trajectory embeddings. *arXiv preprint arXiv:2207.14539*.

Liu, K., *et al.*, 2022. Modeling trajectories with multi-task learning. *In*: *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, Paphos, Cyprus. IEEE, 208–213. https://doi.org/10.1109/MDM55031.2022.00049.

Liu, Q., *et al.*, 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. *In*: *Proceedings of the 30th conference on artificial intelligence (AAAI 2016)*. Arizona, United States: AAAI Press, 194–200.

Liu, Y., Kwan, M.P., and Yu, C., 2023. The uncertain geographic context problem (ugcop) in measuring people's exposure to green space using the integrated 3s approach. *Urban Forestry & Urban Greening*, 85, 127972.

Luca, M., *et al.*, 2021. A survey on deep learning for human mobility. *ACM Computing Surveys*, 55 (1), 1–44.

Mao, Z., *et al.*, 2022. Jointly contrastive representation learning on road network and trajectory. *In*: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. New York, NY: Association for Computing Machinery, 1501–1510. https://doi.org/10.1145/3511808.3557370.

Messaoud, K., *et al.*, 2021. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. *In: 2021 IEEE Intelligent Vehicles Symposium (IV)*, Nagoya, Japan. IEEE, 165–170. https://doi.org/10.1109/IV48863.2021.9576054.

Perozzi, B., Al-Rfou, R., and Skiena, S., 2014. Deepwalk: Online learning of social representations. *In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY: Association for Computing Machinery, 701–710. https://doi.org/10.1145/2623330.2623732.

Qian, C., *et al.*, 2019. Vehicle trajectory modelling with consideration of distant neighbouring dependencies for destination prediction. *International Journal of Geographical Information Science*, 33 (10), 2011–2032.

Rathore, P., *et al.*, 2019. A scalable framework for trajectory prediction. *IEEE Transactions on Intelligent Transportation Systems*, 20 (10), 3860–3874.

Robertson, C., and Feick, R., 2018. Inference and analysis across spatial supports in the big data era: Uncertain point observations and geographic contexts. *Transactions in GIS*, 22 (2), 455–476.

Rudenko, A., *et al.*, 2019. Human motion trajectory prediction: A survey.

Schreier, M., Willert, V., and Adamy, J., 2014. Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems. *In: 17th international ieee conference on intelligent transportation systems (ITSC)*, Qingdao, China. IEEE, 334–341. https://doi.org/10.1109/ITSC.2014.6957713.

Shmool, J.L., *et al.*, 2018. Developing a gis-based online survey instrument to elicit perceived neighborhood geographies to address the uncertain geographic context problem. *The Professional Geographer*, 70 (3), 423–433.

Siła-Nowicka, K., *et al.*, 2016. Analysis of human mobility patterns from gps trajectories and contextual information. *International Journal of Geographical Information Science*, 30 (5), 881–906.
Sousa, R.S.D., Boukerche, A., and Loureiro, A.A., 2020. Vehicle trajectory similarity: models, methods, and applications. *ACM Computing Surveys*, 53 (5), 1–32.
Tobler, W.R., 1970. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46 (sup1), 234–240.
Vaswani, A., *et al.*, 2017. Attention is all you need. *In: Advances in Neural Information Processing Systems*. Curran Associates, Inc., 30.
Veličković, P., *et al.*, 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.
Wan, H., *et al.*, 2022. Pre-training time-aware location embeddings from spatial-temporal trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 34 (11), 5510–5523.
Wu, H., *et al.*, 2023. Online map-matching assisted by object-based classification of driving scenario. *International Journal of Geographical Information Science*, 37 (8), 1872–1907.
Wu, H., *et al.*, 2017. Modeling trajectories with recurrent neural networks. IJCAI.
Wu, R., *et al.*, 2018. Location prediction on trajectory data: A review. *Big Data Mining and Analytics*, 1 (2), 108–127.
Xu, J., *et al.*, 2021. Predicting destinations by a deep learning based approach. *IEEE Transactions on Knowledge and Data Engineering*, 33 (2), 651–666.
Yang, C., and Gidofalvi, G., 2018. Fast map matching, an algorithm integrating hidden markov model with precomputation. *International Journal of Geographical Information Science*, 32 (3), 547–570.
Yang, P., *et al.*, 2021. T3s: Effective representation learning for trajectory similarity computation. *In*: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, Chania, Greece. IEEE, 2183–2188. https://doi.org/10.1109/ICDE51399.2021.00221.
Yao, D., *et al.*, 2019. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. *In: 2019 IEEE 35th international conference on data engineering (ICDE)*, Macao, China. IEEE, 1358–1369. https://doi.org/10.1109/ICDE.2019.00123.
Yao, D., *et al.*, 2017. Trajectory clustering via deep representation learning. *In*: *2017 international joint conference on neural networks (IJCNN)*, Anchorage, AK, USA. IEEE, 3880–3887. https://doi.org/10.1109/IJCNN.2017.7966345.
Ye, N., *et al.*, 2016. Vehicle trajectory prediction based on hidden markov model.
Ying, H., *et al.*, 2019. Time-aware metric embedding with asymmetric projection for successive poi recommendation. *World Wide Web*, 22 (5), 2209–2224.
Yu, W., Zhang, Y., and Chen, Z., 2019. Automated generalization of facility points-of-interest with service area delimitation. *IEEE Access.*, 7, 63921–63935.
Yu, W., *et al.*, 2022. Sparse reconstruction with spatial structures to automatically determine neighbors. *International Journal of Geographical Information Science*, 36 (2), 338–359.
Zhang, Y., Yu, W., and Chen, Z., 2022. An improved method for generalisation of point features with consideration of reinforcing relationships. *Journal of Spatial Science*, 67 (1), 41–60.
Zhao, P., Kwan, M.P., and Zhou, S., 2018. The uncertain geographic context problem in the analysis of the relationships between obesity and the built environment in Guangzhou. *International Journal of Environmental Research and Public Health*, 15 (2), 308.
Zhu, D., and Liu, Y., 2017. An incremental map-matching method based on road network topology. *Geomatics and Information Science of Wuhan University*, 42 (1), 77–83.

## Appendix A. Details of the transformer encoder layer

The Transformer encoder layer consists of a multi-head attention sub-layer (*MultiHead*) and a feed-forward sub-layer (*FFN*), each accompanied by a residual connection. After each of the two sub-layers, layer normalization (*LayerNorm*) (Ba *et al*. 2016) is applied. In other words, the output of each sub-layer is calculated as $LayerNorm(x + SubLayer(x))$, where *SubLayer* represents a function that can be either *MultiHead* or *FFN*. In the Transformer encoder layer, position

embedding (*PE*) is firstly added to the trajectory representation $h_t^{(0)}$ to obtain the the initial input.

$$\begin{aligned} h_t^{(0)} &= PE(h_t^{(0)}) \\ \bar{h}_t^{(l-1)} &= LayerNorm\left(h_t^{(l-1)} + MultiHead(h_t^{(l-1)})\right) \\ z_t^{(l)} &= LayerNorm\left(\bar{h}_t^{(l-1)} + FFN(\bar{h}_t^{(l-1)})\right) \end{aligned} \tag{A1}$$

where *PE* represents the positional embedding for the points in a trajectory, indicating their corresponding sequential order.

## A.1. Multi-head attention

Given a sequence input $X \in \mathbb{R}^{L\times d}$ where $L$ is the length of the sequence and $d$ is feature dimension, multi-head attention is given by:

$$\begin{aligned} \text{MultiHead}\ (\text{Q},\text{K},\text{V}) &= (||_{i=1}^{h} head_i)W^O \\ head_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \\ \text{Attention}\ (\text{Q},\text{K},\text{V}) &= \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \end{aligned} \tag{A2}$$

Here, $Q$, $K$, and $V$ denote the query, the keys, and the values for the input $X$, respectively. $W_i^Q$, $W_i^K$, and $W_i^V$ are the learnable parameters of the $i$th attention head for the query, the keys, and the values, respectively. $W^O$ is a learnable parameter for the output, and $h$ denotes the number of attention heads. $||_{i=1}^{h}$ denotes a successive concatenation operation that concatenates the output of $h$ attention heads.

## A.2. Feed-forward

The feed-forward network consists of a fully connected layer with a ReLU activation function:

$$FFN(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{A3}$$

where $W_1$ and $W_2$ are two learnable weights, and $b_1$ and $b_2$ are the biases for the layer.

# Appendix B. Details of graph forward

After fusing global and local features (i.e., $Z_i^{(l)} \in \mathbb{R}^{|V_i|\times d}$), a graph forward sub-layer is designed for graph neighboring information propagation. In this layer, the GAT (Veličković *et al.* 2017) model is adopted. With this sub-layer, extracted topology information of neighboring roads (i.e., subgraph structure) can be used for graph node embedding updating.

$$\begin{aligned} \overrightarrow{Z}_{i,j}^{(l)} &= ||_{m=1}^{h} LeakyReLU\Big(\sum_{k\in\mathcal{N}_{i,j}} a_{i,j,k,m} W_1^m Z_{i,k}^{(l)}\Big) \\ a_{i,j,k,m} &= \frac{\exp\left(LeakyReLU(\partial^m\left[W_2^m Z_{i,j}^{(l)}, W_2^m Z_{i,k}^{(l)}\right])\right)}{\sum_{k\in\mathcal{N}_{i,j}} \exp\left(LeakyReLU(\partial^m\left[W_2^m Z_{i,j}^{(l)}, W_2^m Z_{i,k}^{(l)}\right])\right)} \end{aligned} \tag{B1}$$

where $Z_{i,j}^{(l)}$ denotes the feature of the $j$ – *th* road in the subgraph $G_i$, $\mathcal{N}_{i,j}$ is the neighboring roads that have direct connections to it, $h$ is the number of heads, $[,]$ denotes concatenation operation, $\partial^m$, $W_1^m$, $W_2^m$ are learnable parameters, $\overrightarrow{Z}_{i,j}^{(l)} \in \mathbb{R}^{|V_i|\times d}$ is the output. $||_{m=1}^{h}$ denotes a successive concatenation operation that concatenates the output of $h$ attention heads.

## Appendix C. Details of graph norm

A graph normalization layer (Dwivedi and Bresson 2020) specifically designed for graph is used based on the output of graph fusion or graph forward. In this layer, given a min-batch of graphs $\{G_{t1}, G_{t2}, ..., G_{tb}\}$ and corresponding graph features $\left\{Z_{t1}^{(l)}, Z_{t2}^{(l)}, ..., Z_{tb}^{(l)}\right\}$, where $Z_{ti}^{(l)} = \{Z_{ti,1}^{(l)}, Z_{ti,2}^{(l)}, ..., Z_{ti,\ell}^{(l)}\}$, $b$ is the batch size and $l$ denotes the point count of each trajectory $ti$. Firstly, a mean pooling operation is performed to obtain graph feature for each subgraph.

$$M_{ti,j}^{(l)} = \frac{1}{|V_{ti,j}|} \sum_{k=1}^{|V_{ti,j}|} Z_{ti,j,k}^{(l)} \tag{C1}$$

where $\left|V_{ti,j}\right|$ is the road (or node) count of the subgraph $G^{ti,j}$. $M^{(l)} = \{M_{ti,j}^{(l)}\}$ where $i \in \{1, 2, ...b\}$ and $j \in \{1, 2, ...\ell\}$, can be obtained. Finally, batch normalization on $M^{(l)} \in \mathbb{R}^{b \times \ell \times d}$ are performed to obtain the final representation $Z\_norm_{ti}^{(l)}$.

$$\begin{aligned} \mu_b &= \frac{1}{b * \ell} \sum_{i=1}^{b} \sum_{j=1}^{\ell} M_{ti,j}^{(l)} \\ \sigma_b &= \frac{\sum_{i=1}^{b} \sum_{j=1}^{\ell} \sum_{k=1}^{|V_{ti,j}|} \left(Z_{ti,j,k}^{(l)} - \mu_b\right)^2}{\sum_{i=1}^{b} \sum_{j=1}^{\ell} |V_{ti,j}|} \\ Z\_norm_{ti}^{(l)} &= \alpha_b \frac{Z_{ti}^{(l)} - \mu_b}{\sqrt{\sigma_b + \varepsilon}} + \beta_b \end{aligned} \tag{C2}$$

where $\alpha_b$ and $\beta_b$ are learnable parameters.

## Appendix D. Details of LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem associated with traditional RNNs. LSTM networks are particularly well-suited for modeling sequential data and time-series information. The key innovation of LSTMs lies in their ability to maintain a memory cell, allowing them to capture long-range dependencies in the input sequence. This is achieved through a set of gating mechanisms, including input, forget, and output gates, which regulate the flow of information into and out of the memory cell. These gates enable LSTMs to selectively remember or forget information over varying time scales, making them effective in tasks such as NLP, speech recognition, and time-series prediction. LSTM can be described as follows:

$$\begin{aligned} \Gamma_f^{(t)} &= \sigma(W_f[h^{t-1}, x^t] + b_f) \\ \Gamma_u^{(t)} &= \sigma(W_u[h^{t-1}, x^t] + b_u) \\ \hat{c}^{(t)} &= \tanh\left(W_c[h^{t-1}, x^t] + b_c\right) \\ c^{(t)} &= \Gamma_f^{(t)} * c^{(t-1)} + \Gamma_u^{(t)} * \hat{c}^{(t)} \\ \Gamma_o^{(t)} &= \sigma(W_o[h^{t-1}, x^t] + b_o) \\ h^t &= \Gamma_o^{(t)} * \tanh(c^{(t)}) \end{aligned} \tag{D1}$$

where $\Gamma_f^{(t)}$, $\Gamma_u^{(t)}$, $\Gamma_o^{(t)}$ denote the forget gate, the update gate, and the output gate, respectively. $h^{(t-1)}$ is the output of the last cell, $x^t$ is the input of this cell, and $h^t$ is the output.