GraphMM: Graph-Based Vehicular Map Matching by Leveraging Trajectory and Road Correlations

Yu Liu, Qian Ge, Wei Luo, Qiang Huang, Lei Zou, Haixu Wang, Xin Li, and Chang Liu

***Abstract*—Map matching of sparse vehicle trajectories is a fundamental problem in location-based services, such as traffic flow analysis and vehicle routing. Existing literature mainly relies on sequence-to-sequence (Seq2Seq) models to capture the *intra-trajectory correlation* of an input trajectory and to sequentially predict the matched road segments. Due to the limited expressive capability of sequential models, these methods fall short of extracting *inter-trajectory* and *trajectory-road correlations* as well as *correlation between road segments*. We present GraphMM, a graph-based approach that explicitly utilizes all aforementioned correlations. Our model exploits the graph nature of map matching and incorporates graph neural networks and conditional models to leverage both road and trajectory graph topology, while manages to align road segments and trajectories in latent space. We formally analyze the expressive power of our model in capturing various correlations and propose efficient algorithms for model training and inference. In particular, our optimization techniques dramatically reduce the computational complexity, making our model feasible on datasets with thousands of road segments. Extensive experiments show that our model significantly enhances prediction accuracy, while improving training and inference efficiency by up to an order of magnitude over both the industrial implementation of the hidden Markov model and state-of-the-art Seq2Seq-based methods.**

***Index Terms*—Conditional model, graph neural network, inductive capability, map matching, trajectory and road correlations.**

I. Introduction

MAP matching is a fundamental problem in location-based services, which aligns vehicle or human trajectories onto the road network. In particular, *vehicular map matching* is of vital importance for industrial communities. For example, in Tencent Maps, it serves as a critical component for event detection (e.g., road closure), traffic flow analysis, mining of daily commute data, and vehicle routing including navigation and estimated time of arrival (ETA) prediction. As illustrated in Fig. 1, in many scenarios, the trajectory is recorded as a sequence of trajectory points with a low sampling rate [1], [2]. Here, the blue dots connected by dashed arrows represent their GPS locations, which are with low positioning accuracy caused by GPS drifts near high buildings or viaducts. This poses great challenges for *map matching of sparse vehicle trajectories*, i.e., recovering the corresponding matched road segments (shown in red lines) of the trajectory.

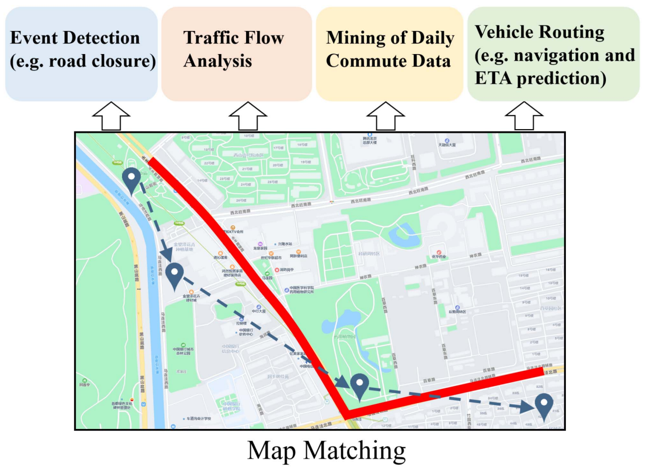

Event Detection (e.g. road closure)
Traffic Flow Analysis
Mining of Daily Commute Data
Vehicle Routing (e.g. navigation and ETA prediction)
Map Matching

Fig. 1. Map matching and its applications in Tencent Maps.

Considerable work has been devoted to map matching for sparse vehicle trajectories. In their classic work [3], Newson and Krumm adopt a hidden Markov model (HMM) by treating trajectories as observations and the matched road segments as states, which outperforms previous geometric matching algorithms. Recently, sequence-to-sequence (Seq2Seq) based methods [4], [5], [6] have achieved state-of-the-art performance. Given the input trajectory that contains a sequence of geopoints, the encoder-decoder framework outputs a sequence of recovered road segments. The accuracy of Seq2Seq-based methods mainly relies on the explicit modeling of *intra-trajectory correlation*. More precisely, each input trajectory is sequentially encoded while the correlations between trajectory points are captured. Nonetheless, we emphasize that the following correlations are equivalently important and should be *explicitly* considered as well.

- *Inter-trajectory correlation:* Trajectories matched to overlapping sequences of road segments have inter-trajectory correlation. This information can be exploited by constructing a trajectory graph from the set of input trajectories by leveraging the well-adopted grid representation [4], [7],

Manuscript received 22 November 2022; revised 11 May 2023; accepted 1 June 2023. Date of publication 20 June 2023; date of current version 27 November 2023. This work was supported in part by 2022 Tencent Big Travel Rhino-Bird Special Research Program and in part by NSFC under Grants 62202037, 61932001, and U20A20174. Recommended for acceptance by R. Chi-Wing Wong. *(Yu Liu and Qian Ge contributed equally to this work.) (Corresponding author: Qiang Huang.)*

Yu Liu, Wei Luo, and Haixu Wang are with Beijing Jiaotong University, Beijing 100044, China (e-mail: yul@bjtu.edu.cn; weiluo@bjtu.edu.cn; import@bjtu.edu.cn).

Qian Ge and Lei Zou are with Peking University, Beijing 100871, China (e-mail: geqian@pku.edu.cn; zoulei@pku.edu.cn).

Qiang Huang, Xin Li, and Chang Liu are with Tencent Maps, Beijing 100193, China (e-mail: johnnhuang@tencent.com; clarkxinli@tencent.com; levenliu@tencent.com).

Digital Object Identifier 10.1109/TKDE.2023.3287739

[8]. *In this way, a trajectory can calibrate its matching via nearby trajectory points in the trajectory graph, possibly from other similar trajectories.*

- *Trajectory-road correlation:* There exists a natural correlation between trajectories and road segments, since a trajectory represents the chronologically travelled road segments of a vehicle. Therefore, *it is reasonable for the model to align a trajectory with its matched road segments, for example, in latent space*.
- *Correlation between road segments:* A vehicle trajectory should be matched to a sequence of *consecutive* road segments. We can model all road segments as a *road graph*, where each node represents a road segment. An edge is established between two nodes if the corresponding road segments are connected. Even for a low-sampling-rate trajectory, *two consecutive trajectory points should be matched to road segments that are reachable in the road graph*. In other words, the correlation of predicted road segments should also be considered.

We note that existing HMM-based and Seq2Seq-based methods are unable to fully exploit these correlations due to limited expressive capability of sequential models, which is formally proved in Section III. Motivated by the above observations, we propose a graph-based approach to explicitly model *all aforementioned correlations* and make the following contributions.

Model Framework. We present **GraphMM**, a Graph-based framework for Map Matching of vehicle trajectories. We construct two graphs, namely, the *road graph* that contains all road segments and the *trajectory graph* that incorporates the input trajectories. Our model heavily exploits the graph nature of the map matching problem. To the best of our knowledge, **GraphMM** is the first map matching framework in which graph plays a central role. Besides, the necessity of capturing above correlations as well as the expressive power of our method are formally proved. Generally speaking, inter-trajectory correlation and the correlation between road segments enhance the discriminatory power of latent representation, whereas trajectory-road correlation facilitates the alignment of representations in latent space.

Graph-Augmented Trajectory Encoder: We leverage the expressive power of graph neural networks (GNN) to enhance the features of trajectory points. Specifically, we use a trajectory graph convolution layer to explicitly integrate information (i.e., *inter-trajectory correlation*) from a set of trajectories. Partially inspired by the success of label propagation (e.g., [9], [10]), we additionally propose a road graph convolution layer to augment the feature of trajectory points by encoding nearby roads. Then, the whole trajectory is fed into the trajectory representation layer (e.g., an RNN-based encoder) where *intra-trajectory correlation* is extracted.

Trajectory-Road Alignment With Inductive Capability: Given the representation of the input trajectory, the decoder recovers the sequence of matched road segments. To explicitly capture *trajectory-road correlation*, the *hidden similarity computation layer* is introduced. With the help of the road graph convolution layer, we are able to compute road segment representation. In each step of the decoder, we align its hidden representation and those of road segments by matrix-vector multiplication, which outputs the matching probability distribution over all road segments. The advantages are three fold. First, we avoid the fully connected layer (i.e., a large learnable transformation matrix) in Seq2Seq-based methods and the overfitting problem. Consequently, our approach significantly enhances model efficiency. Lastly, it guarantees inductive capability, namely, our model does not necessarily need to train from scratch when the road network changes.

Graph-Based Conditional Decoder: We further apply a *conditional layer* based on the conditional random field (CRF) for an extensive exploitation of *correlation between the predicted road segments*. We leverage this correlation (i.e., the road graph) to define the pairwise potential, which improves prediction accuracy and reduces the number of learnable parameters. Intuitively, this layer enables us to firstly predict the easier part of the trajectory (e.g., straight roads) and then tackle the difficult part (e.g., crossroads) by using previous prediction for calibration.

Efficient Model Training and Inference: It is non-trivial to integrate all aforementioned components while ensuring the computational tractability of the model. We present efficient algorithms to facilitate model training and inference. We first design a mini-batch training algorithm to integrate the encoder-decoder framework with graph neural networks and conditional models. We further apply approximation and pruning techniques to ensure model scalability on real-world problem sizes. Besides, we conduct a detailed complexity analysis in terms of model parameters as well as training and inference complexity. We show that **GraphMM** achieves a balance between the HMM-based method and Seq2Seq-based methods in terms of model parameters, but indeed has the most expressive capability for map matching. We conduct extensive experiments on the Tencent dataset for both transductive and inductive settings, which contains tens of thousand trajectories and a road network of more than eight thousand road segments. The dataset is released for public research. Empirical results demonstrate that compared to the best baselines, our approach significantly enhances prediction accuracy while improves model efficiency by up to an order of magnitude, and demonstrates inductive capability. Our source code and data have been made available at https://github.com/GraphMMmaster/GraphMM-Master.

II. Preliminaries

A. Problem Definition

Definition 1 (Road Graph): We use a directed graph $\mathcal{G}_\mathcal{R} = (\mathcal{V}_\mathcal{R}, \mathcal{E}_\mathcal{R})$ to represent the road network. Each node $u_i \in \mathcal{V}_\mathcal{R}$ represents a road segment, while each edge $e_{ij} = (u_i, u_j) \in \mathcal{E}_\mathcal{R}$ represents the intersection of road segment u_i and u_j, satisfying that u_i precedes u_j. The feature of node u_i is the starting and ending GPS location of the corresponding road segment.

Definition 2 (Trajectory): A trajectory $\mathcal{T}$ is defined as a chronological sequence of trajectory point, i.e., $\mathcal{T} = (p_1, \ldots, p_l)$. Each *trajectory point* (a.k.a. geopoint) $p_i \in \mathcal{T}$ is represented by its GPS location (lat_i, lng_i) and the corresponding timestamp t_i, where lat_i (resp. lng_i) stands for latitude (resp. longitude).

TABLE I
TABLE OF NOTATIONS

Notation	Description
$\mathcal{G}_\mathcal{R} = (\mathcal{V}_\mathcal{R}, \mathcal{E}_\mathcal{R})$	the road graph, where each node represents a road segment, we have $n_\mathcal{R} = \lvert\mathcal{V}_\mathcal{R}\rvert$ and $m_\mathcal{R} = \lvert\mathcal{E}_\mathcal{R}\rvert$
$\mathcal{T} = (p_1, \dots, p_l)$	a trajectory, and each p_i is a trajectory point
$\mathcal{G}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$	the trajectory graph constructed from a set of trajectories $\{\mathcal{T}_1, \dots, \mathcal{T}_S\}$ $(n_\mathcal{T}, m_\mathcal{T} = \lvert\mathcal{V}_\mathcal{T}\rvert, \lvert\mathcal{E}_\mathcal{T}\rvert)$
u, v	a road node in $\mathcal{G}_\mathcal{R}$ and a trajectory node in $\mathcal{G}_\mathcal{T}$, i.e., a grid
$\mathbf{A}_\mathcal{R}, \mathbf{A}_\mathcal{T}$	the adjacency matrix of $\mathcal{G}_\mathcal{R}$ (resp. $\mathcal{G}_\mathcal{T}$)
$\mathbf{h}_u, \mathbf{h}_v, \mathbf{h}_p$	the hidden representation of road node u, trajectory node v, and geopoint p
$\mathbf{h}_\mathcal{T}$	the hidden representation of trajectory $\mathcal{T}$
$\mathbf{h}_i^E, \mathbf{h}_i^D$	the hidden representation at the i-th step of the sequential encoder (resp. decoder)
$\mathbf{y}_i, \hat{\mathbf{y}}_i, \tilde{\mathbf{y}}_i$	the ground truth vector and predicted vectors at the i-th step of the decoder
$u_i, \hat{u}_i$	the ground truth and predicted road segment at the i-th step of the decoder

For many real-world applications, trajectory points are sampled with fixed time interval ε, which is referred to as the *ε-sampling-rate trajectory*. Specifically, when ε is sufficiently large (e.g., 30 seconds), the trajectory is called a *sparse trajectory* (a.k.a. *low-sampling-rate trajectory*). We present the formal definition of our problem as below following existing studies [4], [5], [6].

Definition 3 (Map Matching of Sparse Vehicle Trajectories): Given the road graph $\mathcal{G}_\mathcal{R}$ and a set of sparse vehicular trajectories $\{\mathcal{T}_1, \dots, \mathcal{T}_S\}$, for each trajectory $\mathcal{T} = (p_1, \dots, p_l)$, map it to a sequence of nodes in $\mathcal{G}_\mathcal{R}$, denoted as $\mathcal{R} = (u_1, \dots, u_{l'})$. We refer to $\mathcal{R}$ as the *map-matched trajectory* (a.k.a. the route), which is essentially a sequence of matched road segments, where the concatenation of all road segments gives the calibrated trajectory.

Table I lists the notations that are frequently used in the remainder of the paper. Note that for simplicity of notation, throughout this paper we assume all hidden representations are of dimension d.

B. Graph Neural Networks (GNN)

Substantial progress has been made in various graph learning tasks such as node classification and link prediction with the advance of graph convolutional neural networks, for both *transductive* and *inductive* settings [11], [12]. Generally speaking, GNN models compute a d-dimensional representation for each node to encode the information of nearby graph structure and node (and edge) features by conducting multiple layers of graph convolution. From the spatial perspective, for each node u, the graph convolution layer aggregates information of u's neighborhood $N(u)$ to update the representation of u. Most spatial-based GNN models can be *generalized* as the following message-passing framework:

$$\mathbf{m}_u^{(l+1)} = \text{AGGREGATE}^{(l)}\left(\{\mathbf{h}_v^{(l)}, v \in N(u)\}\right), \quad (1)$$

$$\mathbf{h}_u^{(l+1)} = \text{COMBINE}^{(l+1)}\left(\mathbf{h}_u^{(l)}, \mathbf{m}_u^{(l+1)}\right), \quad (2)$$

where $\mathbf{h}_u^{(l)} \in \mathbb{R}^d$ denotes the latent representation of u at the l-th layer, whereas the AGGREGATE and COMBINE operators can be implemented by various mechanisms. In particular, classical GNN models such as the graph convolutional network (GCN) [13] can be *simplified* as follows:

$$\mathbf{h}_u^{(l+1)} = \sigma\left(\sum_{v \in N(u) \cup \{u\}} \mathbf{P}_{uv} \mathbf{h}_v^{(l)} \mathbf{W}^{(l)}\right), \quad (3)$$

where $\mathbf{P}_{uv} \in \mathbb{R}$ denotes the influence of v to u, $\mathbf{W}^{(l)} \in \mathbb{R}^{d \times d}$ denotes the learnable parameters at the l-th layer, and $\sigma(\cdot)$ denotes the activation function such as ReLU. The propagation matrix $\mathbf{P}$ is commonly implemented as a function of the adjacency matrix $\mathbf{A}$. For example, GCN sets the propagation matrix as $\mathbf{P} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\tilde{\mathbf{D}} = \tilde{\mathbf{A}}\mathbf{1}$ is the diagonal matrix of $\tilde{\mathbf{A}}$ containing node degrees.

Recent literature has studied GNN's expressive power (e.g., [14], [15], [16]), deeper GNN models (e.g., [17], [18]), and model scalability (e.g., [19], [20]). Specifically, the Graph Isomorphism Network (GIN) [14] is proved to be as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test [21]. GIN adopts the following equation for message passing:

$$\mathbf{h}_u^{(l+1)} = \text{MLP}^{(l)}\left((1 + \varepsilon^{(l)}) \cdot \mathbf{h}_u^{(l)} + \sum_{v \in N(u)} \mathbf{h}_v^{(l)}\right), \quad (4)$$

where $\text{MLP}^{(l)}$ denotes a multi-layer perceptron and $\varepsilon^{(l)}$ denotes a learnable parameter at l-th layer.

III. GRAPH-BASED MAP MATCHING (GRAPHMM)

A. Model Overview

In this section, we present Graph-based Map Matching (**GraphMM**), which leverages graph structure to model trajectory and road correlations. As shown in Fig. 2, **GraphMM** follows the encoder-decoder framework to map an input trajectory $\mathcal{T}$ to the map-matched trajectory $\mathcal{R}$, i.e., a sequence of road segments. In particular, the *graph-augmented trajectory encoder* takes the road graph $\mathcal{G}_\mathcal{R}$, the constructed trajectory graph $\mathcal{G}_\mathcal{T}$ (for more details see below), and one trajectory $\mathcal{T}$ as input and computes the trajectory representation $\mathbf{h}_\mathcal{T}$ for $\mathcal{T}$. The *graph-based conditional decoder* first uses a sequential decoder to transform $\mathbf{h}_\mathcal{T}$ into a sequence of matched road segments, where the hidden similarity computation layer plays the central role. Next, the prediction can be optionally refined by the conditional layer, which explicitly leverages correlation underlying the road graph structure.

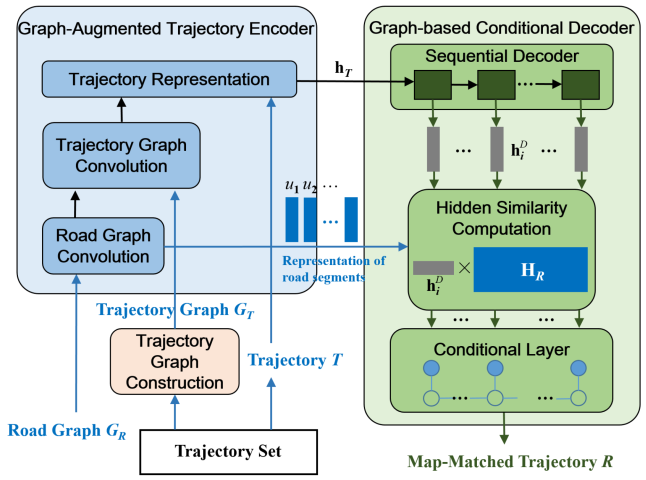

Graph-Augmented Trajectory Encoder
Trajectory Representation
Trajectory Graph Convolution
Road Graph Convolution
Trajectory Graph G_T
Trajectory Graph Construction
Trajectory T
Road Graph G_R
Trajectory Set
Representation of road segments
Graph-based Conditional Decoder
Sequential Decoder
Hidden Similarity Computation
H_R
Conditional Layer
Map-Matched Trajectory R

Fig. 2. Overview of GraphMM.

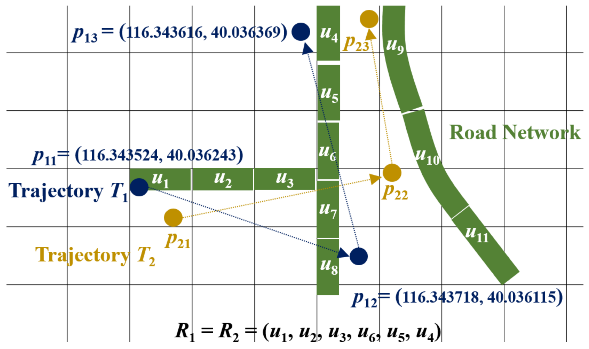

p_{13} = (116.343616, 40.036369)
p_{11} = (116.343524, 40.036243)
p_{12} = (116.343718, 40.036115)
Trajectory T_1
Trajectory T_2
Road Network
$R_1 = R_2 = (u_1, u_2, u_3, u_6, u_5, u_4)$

Fig. 3. Toy example showing grids, trajectories, and road segments.

B. Graph-Augmented Trajectory Encoder

The graph-augmented trajectory encoder contains a *trajectory graph convolution layer* which operates on the trajectory graph, followed by the sequential *trajectory representation layer* and preceded by the *road graph convolution layer* which augments the feature of each geopoint with road information (we postpone this layer to the end of Section III-B). Since the encoder depends on the trajectory graph, we first describe the *trajectory graph construction module*.

1) Trajectory Graph Construction: Given a set of input trajectories,[1] we construct a trajectory graph by leveraging the well-adopted grid representation [4], [7], [22]. Specifically, the whole area of the map is divided into a two dimensional array of grids, where each grid g_i is represented by its index (x_i, y_i). Note that every trajectory point p falls into some grid, denoted as $Grid(p)$.

Definition 4 (Trajectory Graph): Given a set of trajectories $\{\mathcal{T}_1, \ldots, \mathcal{T}_S\}$, the trajectory graph $\mathcal{G}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ represents the union of all trajectories. The node set $\mathcal{V}_\mathcal{T}$ is defined as $\mathcal{V}_\mathcal{T} = \cup_{k=1}^{S}\{Grid(p_i)|p_i \in \mathcal{T}_k\}$, where $Grid(\cdot)$ is a function that maps each p_i to the grid it falls into. The edge set $\mathcal{E}_\mathcal{T}$ contains directed edges which can be associated with edge weights. For each edge $e_{ij} = (v_i, v_j, w_{ij})$, we can set $w_{ij} = |\{(p_x, p_{x+1})|Grid(p_x) = v_i, Grid(p_{x+1}) = v_j, (p_x, p_{x+1}) \in \mathcal{T}_k, k \in \{1, \ldots, S\}\}|$. With some abuse of notation, $(p_x, p_{x+1}) \in \mathcal{T}_k$ means that p_x and p_{x+1} are two consecutive geopoints of $\mathcal{T}_k$.

Note that *the trajectory graph is essentially defined on the grids, but only consider those having intersection with trajectories*. It can be constructed in linear time and space complexity with respect to the trajectory set. More precisely, both complexity is bounded by $O(\sum_{k=1}^{S}|\mathcal{T}_k|)$, where $|\mathcal{T}|$ is the number of geopoints in $\mathcal{T}$. The trajectory graph provides an *explicit* way to take the advantage of *inter-trajectory correlations*, as opposed to existing methods that implicitly leverage the correlation via directly feeding trajectories into sequential models.

Example 1: Consider the toy example in Fig. 3 . We have two low-sampling-rate trajectories $\mathcal{T}_1 = (p_{11}, p_{12}, p_{13})$ (in dark blue) and $\mathcal{T}_2 = (p_{21}, p_{22}, p_{23})$ (in brown) and eleven road segments from u_1 to u_{11}. For $\mathcal{T}_1$ we also show the GPS position of each trajectory point. Both trajectories are matched to same sequences of road segments $(u_1, u_2, u_3, u_6, u_5, u_4)$. It can be seen that the GPS drift of trajectory points p_{22} and p_{23} might cause erroneous prediction. The trajectory graph $\mathcal{G}_\mathcal{T}$ contains five nodes $\{v_1, \ldots, v_5\}$, with $Grid(p_{11}) = Grid(p_{21}) = v_1$, $Grid(p_{12}) = v_2$, $Grid(p_{13}) = v_3$, $Grid(p_{22}) = v_4$, and $Grid(p_{23}) = v_5$. The edge set is $\{(v_1, v_2), (v_1, v_4), (v_2, v_3), (v_4, v_5)\}$. Note that all edges are directed and are of weight one. Since p_{11} and p_{21} are mapped to the same node, p_{12} and p_{22} (resp. p_{13} and p_{23}) become 2-hop (resp. 4-hop) reachable and now it is possible to calibrate the prediction with the information from other trajectories.

2) Trajectory Graph Convolution: Given the trajectory graph $\mathcal{G}_\mathcal{T}$ as input, the trajectory graph convolution layer can compute the hidden representation for each node in $\mathcal{G}_\mathcal{T}$. For simplicity, we employ a natural extension of the classic graph convolutional network (GCN) [13]. For each node v in $\mathcal{G}_\mathcal{T}$, let $\mathbf{h}_v^{(0)} \in \mathbb{R}^d$ denote its initial feature. For example, we can pass the index of the corresponding grid through a fully connected layer to get $\mathbf{h}_v^{(0)}$. Next, we adopt the graph convolution defined by (3), but change it slightly to consider edge weights. We conduct L layers of convolution to get the hidden representation of a trajectory node v (i.e., a grid), which is denoted as $\mathbf{h}_v^{(L)}$.

Next, $\mathbf{h}_v^{(L)}$ is assigned to every geopoint p matched to node v (i.e., with $Grid(p) = v$). In the following we denote by $\mathbf{GCN}_\mathcal{T}^{(L)}(\cdot)$ the L-layer graph convolution conducted on trajectory graph $\mathcal{G}_\mathcal{T}$, and $\mathbf{h}_p$ the hidden representation for geopoint p:

$$\mathbf{h}_p = \mathbf{GCN}_\mathcal{T}^{(L)}(Grid(p)). \tag{5}$$

To this end, each geopoint aggregates neighborhood information from the trajectory it belongs to as well as nearby trajectories. In general, any GNN model applicable to directed and weighted graphs can be used, and more powerful GNNs might result in

[1]By default we adopt the transductive setting, where trajectories in both training set, validation set and test set are used for trajectory graph construction. For the inductive setting, in model training phase we only use the trajectories in the training set. In the model inference phase, we can either fix the trajectory graph or update it on the fly.

better prediction accuracy. The following proposition states that by capturing the inter-trajectory correlation, our encoder has more expressive power in terms of discernibility.

Proposition 1: Given an input trajectory $\mathcal{T} = (p_1, \ldots, p_l)$, its latent representation $(\mathbf{h}_{p_1}, \ldots, \mathbf{h}_{p_l})$ computed by the trajectory graph convolution layer has more discriminatory power than that of $\mathcal{T}$'s input feature.

Proof: Consider two trajectories $\mathcal{T}_1 = (p_{11}, \ldots, p_{1l})$ and $\mathcal{T}_2 = (p_{21}, \ldots, p_{2l})$ with different sequences of matched road segments $\mathcal{R}_1$ and $\mathcal{R}_2$. There exists at least one $i \in [1, l]$ with *discernible* input features, say, $Grid(p_{1i}) \neq Grid(p_{2i})$ or $\|(lat_{1i}, lng_{1i}) - (lat_{2i}, lng_{2i})\| > \varepsilon$, where ε is a predefined distance threshold. The latter case can be transformed to the former by adjusting grid size or shifting the position of grids. Assume that two sequences of latent representations are *indiscernible*, as RNNs cannot guarantee that different input features are *always* transformed to distinct representations (i.e., *injective*). Let $\mathbf{h}^E_{p_{1i}}$ (resp. $\mathbf{h}^E_{p_{2i}}$) denote the latent representation at the i-th step of the encoder for $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) and assume for all $i \in [1, l]$, $\mathbf{h}^E_{p_{1i}} \approx \mathbf{h}^E_{p_{2i}}$ (e.g., $\|\mathbf{h}^E_{p_{1i}} - \mathbf{h}^E_{p_{2i}}\| \leq \varepsilon$ for sufficiently small ε). Consequently, the predicted road segments are identical, i.e., $\hat{\mathcal{R}}_1 = \hat{\mathcal{R}}_2$. With the trajectory graph convolution layer, it is possible to have $\mathbf{h}^E_{p_{1i}} \neq \mathbf{h}^E_{p_{2i}}$ (e.g., $\|\mathbf{h}^E_{p_{1i}} - \mathbf{h}^E_{p_{2i}}\| > \varepsilon$) as we encode different graph structural information for $v_{1i} = Grid(p_{1i})$ and $v_{2i} = Grid(p_{2i})$. More precisely, according to (1), we have

$$\mathbf{h}^{(l+1)}_{v_x} = \text{COMBINE}^{(l+1)}\left(\mathbf{h}^{(l)}_{v_x}, \text{AGGREGATE}^{(l)}\left(\{\mathbf{h}^{(l)}_v\}\right)\right), \tag{6}$$

for $v_x = \{v_{1i}, v_{2i}\}$ and $v \in N(v_x)$. As $v_{1i} \neq v_{2i}$, it is very probably that their computation graphs (i.e., rooted subtrees) are different, and thus $\mathbf{h}^{(L)}_{v_{1i}} \neq \mathbf{h}^{(L)}_{v_{2i}}$. Then $\mathbf{h}^E_{p_{1i}} \neq \mathbf{h}^E_{p_{2i}}$ follows.

Also note that similar analysis has been conducted for other graph problems by adopting GNNs, e.g., [23].

3) Trajectory Representation: Once the trajectory graph convolution layer computes $\mathbf{h}_{p_i}$ for each trajectory point p_i, the trajectory representation layer transforms the input trajectory $\mathcal{T} = (p_1, \ldots, p_l)$ into its vector representation $\mathbf{h}_{\mathcal{T}} \in \mathbb{R}^d$. We integrate RNN models to sequentially encode the trajectory and extract intra-trajectory correlation. Specifically, for each step $i \in [1, l]$ we use bidirectional GRU:

$$\mathbf{h}^E_i = \mathbf{BiGRU}(\mathbf{h}^E_{i-1}, \mathbf{h}_{p_i}), \tag{7}$$

where $\mathbf{h}^E_i$ is the hidden representation at i-th step of the encoder, and we set $\mathbf{h}_{\mathcal{T}} = \mathbf{h}^E_l$. With a slightly abuse of notation, the process is denoted as $\mathbf{h}_{\mathcal{T}} = \mathbf{BiGRU}(\mathbf{h}_{p_1}, \ldots, \mathbf{h}_{p_l})$. The expressive power of sequential encoder in capturing intra-trajectory correlation is guaranteed by the theoretical analysis of RNNs [24].

4) Road Graph Convolution: We propose the road graph convolution layer *prior to* the trajectory graph convolution layer for two purposes: 1) to compute latent representations for road segments which will be utilized in the decoder (see Section III-C) and 2) to augment the features of trajectory nodes with road information.

For each node u of road graph $\mathcal{G}_{\mathcal{R}}$, let $\mathbf{h}^{(0)}_u$ denote its initial feature, e.g., computed from the grids containing the starting and ending GPS locations of the road segment. We conduct a L'-layer graph convolution on $\mathcal{G}_{\mathcal{R}}$ to get u's representation $\mathbf{h}^{(L')}_u$. We adopt GIN [14] (see (4)) for that we tend to learn distinguishable hidden representations for different road segments:[2]

$$\mathbf{h}^{(L')}_u = \mathbf{GIN}^{(L')}_{\mathcal{R}}(u). \tag{8}$$

Feature Augmentation for Trajectory Nodes: To augment the features of nodes (i.e., grids) in trajectory graph $\mathcal{G}_{\mathcal{T}}$, we only process the grids that intersect with road graph $\mathcal{G}_{\mathcal{R}}$. For each grid v, we augment its feature $\mathbf{h}^{(0)}_v$ with the representation of all intersecting road segments, e.g., by taking element-wise average and concatenating it to the original feature:

$$\mathbf{h}^{(0)}_v = \mathbf{Cat}(\mathbf{h}^{(0)}_v, \mathbf{Mean}_{\{u | u \text{ intersects } v\}}\{\mathbf{h}^{(L')}_u\}). \tag{9}$$

The advantages are two fold. First, it facilitates the alignment of the representations of trajectory points and road segments, which will be illustrated in Section III-C1. Second, recall that vehicular map matching is essentially modeled as a multi-label classification problem [3], [4], [5], [6] with the label set being all road segments. In particular, we are partially inspired by recent advances for node classification where simple model [9] explicitly exploiting label propagation outperforms most sophiscated GNNs.

Example 2: We elaborate the effectiveness of road graph convolution by reconsidering the toy example in Fig. 3. Assume that trajectory node p_{12} should be matched to road segment u_6. Unfortunately, as the GPS location of p_{12} is drifted, directly matching it to u_8 in the same grid gives an unreasonable vehicle routing plan. With the road graph convolution layer, the initial feature of p_{12} now contains the information of u_6 and thus it is possible to correct the prediction. Moreover, the additional road information facilitates to match p_{11} and p_{21} to u_1.

We formally demonstrate the necessity of road segment representation for capturing trajectory-road correlation in Section III-C1.

5) Algorithm Description: The whole process of the graph-augmented trajectory encoder is described in Algorithm 1. Given an input trajectory $\mathcal{T}$, for each trajectory point $p_i \in \mathcal{T}$, let v_i be the grid that p_i falls into (Lines 1-2). If v_i has intersection with road graph $G_{\mathcal{R}}$ (Line 3), we first conduct L'-layer road graph convolution for each overlapping road segment u (Lines 4-5) and then use the element-wise average of their hidden representation to augment the initial representation of v_i (Line 6). Otherwise, the initial representation of v_i only includes the encoding of its own features (Lines 7-8). Next, we conduct L-layer trajectory graph convolution for v_i and set the representation of p_i accordingly (Line 9). After we get the hidden representation of every trajectory point p_i, we take them as the input features and use an RNN encoder to compute the trajectory representation $\mathbf{h}_{\mathcal{T}}$ (Lines 10-11).

[2]This layer can be naturally extended to include edge direction information, for example, to conduct graph convolution on both directions and concatenate the hidden representations at each layer. In practice, we might have to ignore road direction because the data is dirty. For the raw datasets used by industrial communities such as the one in our experiments, a non-negligible portion of trajectories do not correspond to a connected path on the directed road graph, and by removing edge direction this problem is almost solved. This is because the information of a small fraction of roads (e.g., side roads) might be problematic.

Algorithm 1: Graph-Augmented Trajectory Encoder.

```
Input: Road graph $\mathcal{G}_\mathcal{R}$; Trajectory graph $\mathcal{G}_\mathcal{T}$; An input
    trajectory $\mathcal{T} = (p_1, \dots, p_l)$
Output: $\mathbf{h}_\mathcal{T}$, the hidden representation of trajectory $\mathcal{T}$
1  for each $p_i \in \mathcal{T}$ do
2      Let $v_i = Grid(p_i)$ denote the grid that $p_i$ falls into;
3      if $v_i$ has intersection with road graph $\mathcal{G}_\mathcal{R}$ then
4          for each road segment $u$ that intersects $v_i$ do
5              $\mathbf{h}_u^{(L')} = \mathbf{GIN}_\mathcal{R}^{(L')}(u)$;
6          $\mathbf{h}_{v_i}^{(0)} = \mathbf{Cat}(\mathbf{h}_{v_i}^{(0)}, \mathbf{mean}\{\mathbf{h}_u^{(L')}\})$;
7      else
8          $\mathbf{h}_{v_i}^{(0)} = \mathbf{Cat}(\mathbf{h}_{v_i}^{(0)}, \mathbf{0})$;
9      $\mathbf{h}_{p_i} = \mathbf{GCN}_\mathcal{T}^{(L)}(v_i)$;
10 $\mathbf{h}_\mathcal{T} = \mathbf{BiGRU}(\mathbf{h}_{p_1}, \dots, \mathbf{h}_{p_l})$;
11 return $\mathbf{h}_\mathcal{T}$;
```

C. Graph-Based Conditional Decoder

The decoder recovers a sequence of matched road segments from the hidden representation of the input trajectory $\mathcal{T}$. We first employ an RNN-based decoder similar to existing works [4], [5], [6], [7], but additionally present the notion of *hidden similarity computation* to improve model accuracy, efficiency and interpretability. To be precise, by exploiting trajectory-road correlation, we manage to align trajectories and road segments in latent space. We further observe that (attentional) RNN decoders cannot fully make use of the correlation between predicted road segments, and put forward the *conditional layer* to calibrate the prediction. The intuition is that *consecutive geopoints in a trajectory (even for a low-sampling-rate one) should be matched to road segments inter-reachable by a few hops in* $G_\mathcal{R}$. Note that the conditional layer is *optional*, which enables the tradeoff between accuracy and efficiency.

1) Hidden Similarity Computation: At each step of the decoder, the hidden similarity computation layer is employed to convert the d-dimensional hidden representation $\mathbf{h}_i^D$ to an $n_\mathcal{R}$-dimensional probability vector $\tilde{\mathbf{y}}_i$, of which the j-th component $\tilde{\mathbf{y}}_{ij}$ indicates the probability of the j-th road segment being matched. In particular, we are motivated by the following observations and substitute the well-adopted fully connected layer [4], [5], [7] by hidden similarity computation.

Motivation. We note that the number of road segments $n_\mathcal{R}$ is large (at least in thousands), thus simply using a fully connected (FC) layer [4], [5], [7] to convert $\mathbf{h}_i^D \in \mathbb{R}^d$ to $\tilde{\mathbf{y}}_i \in \mathbb{R}^{n_\mathcal{R}}$ needs tremendous parameters. We denote by $\mathbf{W}_{\text{FC}}$ the $d \times n_\mathcal{R}$-dimensional learnable matrix. For the Tencent dataset adopted in our experiments, if d is set to 512 following [4], with $n_\mathcal{R} > 8{,}000$, the FC layer contains more than 4 million learnable parameters. Take **MTrajRec** [4] as an example, together with the road segment ID embedding layer that transforms $\tilde{\mathbf{y}}_i$ to the input of the $(i+1)$-th step (the input dimension is set to 128), it consists of nearly 95% of model parameters. However, the number of trajectories are not on par with the parameter size. For the Tencent dataset, we only have about 64 K trajectories. Therefore, we are facing the overfitting problem. Besides, the FC layer lacks interpretability, that is, the meaning of $\mathbf{W}_{\text{FC}}$ is not clear.

Instead, with the help of the road graph convolution layer in Section III-B, we devise a more interpretable solution to convert $\mathbf{h}_i^D$ to $\tilde{\mathbf{y}}_i$. Recall that we can compute the vector representation of each road segment u (i.e., node u of $G_\mathcal{R}$), thus we concatenate all $n_\mathcal{R}$ node representations and get the representation matrix $\mathbf{H}_\mathcal{R} \in \mathbb{R}^{d \times n_\mathcal{R}}$. It is important to note that $\mathbf{H}_\mathcal{R}$ are not learnable parameters; we just reuse the road graph convolution layer. We then multiply $\mathbf{H}_\mathcal{R}$ and $\mathbf{h}_i^D$ to get the $n_\mathcal{R}$-dimensional prediction. Intuitively, if the part of trajectory represented by $\mathbf{h}_i^D$ is to match with road segment u_i, we want the dot product of $\mathbf{h}_i^D$ and the u_i-th column of $\mathbf{H}_\mathcal{R}$ to be large. Our solution contains much fewer parameters and therefore is easier to train. Meanwhile, it also has better interpretability as the alignment is done in latent space rather than relying on distance-based heuristics. The following proposition guarantees that our solution explicitly utilizes the trajectory-road correlation.

Proposition 2: With the hidden similarity computation layer and the road graph convolution layer, we manage to align representations of trajectory points and road segments in latent space.

Proof: Given the representation matrix $\mathbf{H}_\mathcal{R}$ for all road segments and the latent state $\mathbf{h}_i^D$ at i-th step of the decoder, the hidden similarity computation layer (followed by Softmax) outputs predicted probability distribution $\tilde{\mathbf{y}}_i = \text{Softmax}(\mathbf{H}_\mathcal{R}^\intercal \mathbf{h}_i^D)$. Let u_i denote the ground truth road segment for this step, our objective is to maximize $\tilde{\mathbf{y}}_{iu_i}$ and consequently $\mathbf{h}_{u_i}^\intercal \mathbf{h}_i^D$. In other words, the trajectory point and its matched road segment are aligned in latent space. To understand why the road graph convolution layer is adopted to compute $\mathbf{H}_\mathcal{R}$, note that for two road segments u_i and u_j, if we have $\mathbf{h}_{u_i}^{\text{MLP}} \neq \mathbf{h}_{u_j}^{\text{MLP}}$, then $\mathbf{h}_{u_i}^{\text{GIN}} \neq \mathbf{h}_{u_j}^{\text{GIN}}$ but not vice versa. Here $\mathbf{h}_{u_i}^{\text{MLP}}$ and $\mathbf{h}_{u_i}^{\text{GIN}}$ denote the latent representation computed by an MLP and GIN, respectively. Meanwhile, the objective suggests that better discerniblity between road segment representations helps to improve prediction accuracy.

Example 3: Recall our toy example in Fig. 3. Assume two roads (u_6, u_5, u_4) and (u_{11}, u_{10}, u_9) never intersect in the road map. Unfortunately, trajectory points p_{22} and p_{23} of $\mathcal{T}_2$ are closer to road (u_{11}, u_{10}, u_9), even though they should be matched to (u_6, u_5, u_4) for a reasonable vehicle route. Distance-based mask layer (e.g., [4]) filters out road segments that are beyond a predefined distance from p_{21}, p_{22} and p_{23}. Thus, it might falsely prune the ground truth road segments of p_{22} and p_{23}. Moreover, to recover the matched road segments $(\hat{u}_{21}, \hat{u}_a, \hat{u}_{22}, \hat{u}_b, \hat{u}_c, \hat{u}_{23})$ from the low-sampling-rate trajectory $\mathcal{T}_2$ [4], the distance-based mask layer can only be applied to $\hat{u}_{2i}$ corresponding to p_{2i} for $i \in [1, 3]$. In comparison, our model provides a more flexible approach to align all latent representations $(\mathbf{h}_{21}^D, \mathbf{h}_a^D, \mathbf{h}_{22}^D, \mathbf{h}_b^D, \mathbf{h}_c^D, \mathbf{h}_{23}^D)$ with those of the road segments.

Inductive Capability: Another important advantage of our approach is the inductive capability. In particular, our model can handle changes of road network, for example, adding new road segments. Representative methods [4], [5], [6] have to train the model from scratch since the dimensions of the learned parameter matrix need to be updated. The inductive capability of **GraphMM** is guaranteed by the following model design strategy. We employ graph neural networks to compute the representation of road segments and thus road segment ID embedding is excluded. On the other hand, we use the hidden similarity computation layer to replace the fully connected layer. Consequently, our model is able to borrow the inductive capability from GNNs.

2) Conditional Layer: Few approaches have considered the correlation between matched road segments of the input trajectory, of which the importance is illustrated by the following example.

Example 4: Let us consider trajectory $\mathcal{T}_1 = (p_{11}, p_{12}, p_{13})$ in Fig. 3. Whether to match p_{12} to u_6 or u_7 mainly depends its successor p_{13} rather than its precursor p_{11}. In other words, it depends on whether the vehicle turns left or right at the corner.

However, Seq2Seq-based approaches model this correlation *implicitly* by encoding the whole input trajectory, which is proved to have limited expressive power by the following proposition.

Proposition 3: By explicitly considering the correlation between road segments, it is possible to correctly predicted trajectories with indiscernible input features but with different matched road segments (i.e., labels).

Proof: Let us consider two trajectories $\mathcal{T}_1 = (p_{11}, \ldots, p_{1l})$ and $\mathcal{T}_2 = (p_{21}, \ldots, p_{2l})$ with different sequences of matched road segments $\mathcal{R}_1 = (u_{11}, \ldots, u_{1l})$ and $\mathcal{R}_2 = (u_{21}, \ldots, u_{2l})$. We set the trajectories and matched road segments to have the same length l for simplicity. Specifically, there exists at least one i with $u_{1i} \neq u_{2i}$. However, their input features are *indiscernible*, with $p_{1i} \approx p_{2i}$ for all $i \in [1, l]$. This is possible due to GPS drifts for two vehicle trajectories corresponding to nearby roads. Except for the objective function, if the model structure does not explicitly exploit label information from the matched road segments, it cannot make the correct prediction for both $\mathcal{T}_1$ and $\mathcal{T}_2$ since all latent representations are also indiscernible.$\square$

We observe that there might be multiple ways to model the correlation between road segments. Apart from the road graph convolution layer that explicitly incorporates labels, we employ an additional conditional random field (CRF) layer in the decoder. In contrast to the sequential decoder that predicts each road segment only based on its precursors, the conditional layer utilizes the information of the whole trajectory. In particular, it is built on the output of the RNN-based decoder and maximizes the conditional probability $P(\mathbf{Y}|\tilde{\mathbf{Y}})$, where $\tilde{\mathbf{Y}} \in \mathbb{R}^{l' \times n_{\mathcal{R}}}$ are predicted probability vectors of the l' matched road segments by the hidden similarity computation layer, which is taken as the input feature of CRF, and $\mathbf{Y} \in \mathbb{R}^{l' \times n_{\mathcal{R}}}$ denotes the ground truth. More precisely, the i-th row of $\tilde{\mathbf{Y}}$ (resp. $\mathbf{Y}$), denoted as $\tilde{\mathbf{y}}_i$ (resp. $\mathbf{y}_i$), is an $n_{\mathcal{R}}$-dimensional vector representing the predicted (resp. ground truth) matching probability of each road segment. We define the conditional probability as follows:

$$P(\mathbf{Y}|\tilde{\mathbf{Y}}) = \frac{1}{Z(\tilde{\mathbf{Y}})} \exp\left(\sum_{i=1}^{l'} \psi(\mathbf{y}_i, \tilde{\mathbf{y}}_i) + \sum_{i=2}^{l'} \phi(\mathbf{y}_{i-1}, \mathbf{y}_i)\right), \tag{10}$$

where $Z(\tilde{\mathbf{Y}}) = \sum_{\mathbf{Y}'} \exp(\sum_{i=1}^{l'} \psi(\mathbf{y}'_i, \tilde{\mathbf{y}}_i) + \sum_{i=2}^{l'} \phi(\mathbf{y}'_{i-1}, \mathbf{y}'_i))$ is the partition function, which summarizes the energy score of all possible prediction $\mathbf{Y}'$. We use $\psi(\cdot)$ and $\phi(\cdot)$ to denote the unary and pairwise potential respectively. For our problem, the unary potential is simply defined as the predicted probability of the ground truth road segment:

$$\psi(\mathbf{y}_i, \tilde{\mathbf{y}}_i) = \tilde{y}_{iu_i}, \tag{11}$$

where $\tilde{y}_{iu_i}$ represents the u_i-th element of $\tilde{\mathbf{y}}_i$, and u_i is the ground truth road segment.

However, it is non-trivial to define the pairwise potential. Canonical CRF models use a learnable transition matrix, which is inapplicable to our problem for the following reasons. Since the transition matrix has $n_{\mathcal{R}} \times n_{\mathcal{R}}$ parameters, we still face the overfitting problem. Actually, a naive implementation of the pairwise potential renders inferior accuracy compared to totally excluding the CRF layer. Moreover, such an implementation has severe scalability problems. It is very time consuming to train the CRF with thousands of labels.

We adopt the following pairwise potential, which reflects the label correlation of two consecutive road segments:

$$\phi(\mathbf{y}_{i-1}, \mathbf{y}_i) = \text{ReLU}(\mathbf{h}_{u_{i-1}}^\intercal \mathbf{W}_{\text{CRF}} \mathbf{h}_{u_i})(\mathbf{A}_{\mathcal{R}}^k)_{u_{i-1}u_i}, \tag{12}$$

where $\mathbf{h}_{u_i}$ is the hidden representation of road segment u_i, and $\mathbf{W}_{\text{CRF}}$ is a $d \times d$-dimensional parameter matrix. $\mathbf{A}_{\mathcal{R}}^k$ denotes the k-hop transition matrix of $\mathcal{G}_{\mathcal{R}}$ and is defined as

$$(\mathbf{A}_{\mathcal{R}}^k)_{u,v} = \begin{cases} 1 & \text{if u can reach v in k hops,} \\ -\lambda & \text{otherwise.} \end{cases} \tag{13}$$

Here, the hyper-parameter λ is set to 10,000 to penalize those predictions in which two consecutive road segments are unreachable on road network. In this case, the energy score of the whole prediction will approach zero.

Remark: It is worth noticing that the employment of correlation between road segments is not limited to the above approach. For example, we can calculate higher order transition probabilities from the trajectory set and use (12) to capture travel patterns [25] instead of the adjacency info, or add another objective to minimize the gap between $\mathbf{h}_{u_{i-1}}^\intercal \mathbf{h}_{u_i}$ and transition probabilities, which is beyond the scope of this paper.

3) Algorithm Description: The whole process of graph-based conditional decoder is demonstrated in Algorithm 2. Given the hidden representation of trajectory $\mathcal{T}$, we use the attentional GRU as the sequential decoder (Lines 1-3), and conduct hidden similarity computation to get the initial prediction $(\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_{l'})$ (Line 4). It can be further refined by the CRF layer, denoted as $(\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{l'})$ (Line 5) (this step is optional). Finally the algorithm returns the sequence of all matched road segments (Line 6).

IV. Model Training and Optimization

A. Model Training and Inference

Objective Function: We denote by **GraphMM** and **GraphMM w/o CRF** the model variants with and without the CRF layer, respectively. For **GraphMM w/o CRF**, we adopt the cross-entropy loss (Equation 23) as in [4]. For **GraphMM**, we maximize the conditional probability over all training data,

Algorithm 2: Graph-Based Conditional Decoder.

```
Input: $\mathbf{h}_{\mathcal{T}}$, the hidden representation of input trajectory $\mathcal{T}$;
       the number of matched road segments $l'$
Output: the matched road $\mathcal{R}$
1 $\mathbf{h}_0^D = \mathbf{h}_{\mathcal{T}}$;
2 for $i = 1$ to $l'$ do
3     $\mathbf{h}_i^D$ = AttentionalGRU($\mathbf{h}_{i-1}^D$);
4     $\tilde{\mathbf{y}}_i$ = HiddenSimilarityComputation($\mathbf{h}_i^D$);
5 Refine all matched road segments $(\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_{l'})$ via the CRF
  layer and get $(\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{l'})$ (this step is optional);
6 return $\mathcal{R}$, the sequence of all matched road segments
```

Algorithm 3: Mini-Batch Training.

```
Input: training set $Tr$, batch size $B$
1 for each epoch do
2     for each minibatch $\mathcal{B} = \{(\mathcal{T}_1, \mathcal{R}_1), \ldots, (\mathcal{T}_B, \mathcal{R}_B)\}$ do
3         for each data instance $(\mathcal{T}_i, \mathcal{R}_i)$ do
4             for each $p_{ij} \in \mathcal{T}_i$ do
5                 Forward pass for the road graph convolution
                  and trajectory graph convolution layer;
6             Forward pass for the trajectory representation
              layer;
7             Forward and backward pass for the decoder,
              including the RNN layer and the conditional
              layer;
8             Backward pass for the components of the
              encoder, including the RNN layer, the trajectory
              and road graph convolution layer;
9     Update model parameters;
```

which is equivalent to maximize

$$\mathcal{L} = \sum_{(\mathcal{T},\mathcal{R}) \in Tr} \log P(\mathbf{Y}|\tilde{\mathbf{Y}}), \tag{14}$$

where $\mathbf{Y}$ contains the ground truth of matched road segments, and $\tilde{\mathbf{Y}}$ denotes the input to the CRF layer which has the same dimension to $\mathbf{Y}$. More detailed definition of the conditional probability including the unary and pairwise potentials is referred to (10)–(13).

Mini-Batch Training: We use the mini-batch strategy for model training, as illustrated in Algorithm 3. Each minibatch $\mathcal{B}$ composes of B data instances $(\mathcal{T}_i, \mathcal{R}_i), i \in [1, B]$ (Lines 1-2). For each data instance $(\mathcal{T}_i, \mathcal{R}_i)$ and for every trajectory point $p_{ij} \in \mathcal{T}_i$, we first conduct forward pass of the road and trajectory graph convolution layer, respectively (Lines 3-5). In practice, for simplicity of implementation, both GNNs are computed by the full-batch process, i.e., for each minibatch $\mathcal{B}$, we compute the $n_{\mathcal{R}} \times d$-dimensional road segment representation and the $n_{\mathcal{T}} \times d$-dimensional trajectory node representation for $\mathcal{G}_{\mathcal{R}}$ and $\mathcal{G}_{\mathcal{T}}$ respectively. However, empirical study shows that it is sufficiently fast for the datasets tested. Next, we use mini-batch training and run the forward pass on the whole trajectory (Line 6), followed by the forward and backward pass for the decoder (Line 7). We use the backward pass to compute the gradients for the encoder (Line 8), and update model parameters via stochastic gradient descent (Line 9).

Remark: Both the road and the trajectory graph convolution layer can be easily converted to mini-batch training. For each trajectory node p_{ij}, we only fetch its L-hop neighbors in trajectory graph $\mathcal{G}_{\mathcal{T}}$. For each node v in this receptive field, we find all overlapping road segments. For each road segment u, we fetch its L'-hop neighbors in road graph $\mathcal{G}_{\mathcal{R}}$. Then we can invoke Algorithm 1 to compute the hidden representation of p_{ij}. We leave more details to future work.

Model Inference: This stage is quite straightforward as we sequentially invoke Algorithms 1 and 2 to predict the matched road segments of an input trajectory. If the CRF layer is included, we adopt the Viterbi algorithm to compute the final prediction. To ensure model efficiency, we need the following optimization techniques.

B. Optimizations

We propose several optimizations for efficient training and inference of the CRF layer, as the label set usually contains thousands or more road segments.

Approximation of $Z(\tilde{\mathbf{Y}})$: Recall that the computation of $Z(\tilde{\mathbf{Y}})$ in (10) involves all possible $\mathbf{Y}'$. If the map-matched trajectory is of length l', there exist $O(n_{\mathcal{R}}^{l'})$ possible predictions, which are computationally intractable. However, we note that most $\mathbf{Y}'$s are unreasonable, namely, they do not obey the constraint that two consecutive road segments are inter-reachable. Moreover, these predictions are penalized by our design of pairwise potential, and will have minor contribution to $Z(\tilde{\mathbf{Y}})$.

With the mini-batch strategy, we only consider a small subset of all possible $\mathbf{Y}'$. More specifically, for each trajectory $\mathcal{T}$ in a minibatch $\mathcal{B}$, let $\tilde{\mathbf{Y}} = \{\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_{l'}\} \in \mathbb{R}^{l' \times n_{\mathcal{R}}}$ be the input of CRF, i.e., the prediction made by the sequential layer. For each $\tilde{\mathbf{y}}_i \in \mathbb{R}^{n_{\mathcal{R}}}$, we find its top-$r$ component with the largest values (e.g., $r = 2$ or 3), and fetch the corresponding road segments. We also include the ground truth road segments for each index i. Then we use the union of fetched road segments of this minibatch rather than all $n_{\mathcal{R}}$ road segments to compute an approximation of $Z(\tilde{\mathbf{Y}})$. The approximation is generally accurate because we indeed consider the predictions with large energy score for each trajectory. On the other hand, the number of considered road segments is limited and can be easily manipulated. In practice, we manage to reduce the road segments by orders of magnitude.

Pruning Unnecessary Computation by $\mathcal{G}_{\mathcal{R}}$: A straightforward implementation of the Viterbi algorithm incurs $O(ln_{\mathcal{R}}^2)$ time on an l-sized sequence with $n_{\mathcal{R}}$ states, which is very costly for large $n_{\mathcal{R}}$. However, for map matching the state transition is determined by the road graph topology. In particular, assume at step i, the conditional decoder predicts road segment $\hat{u}_i$. For the next step, our pairwise potential effectively prunes most states other than its k-hop neighbors in $G_{\mathcal{R}}$. That is, the remaining states are only the non-zero components of $\mathbf{A}_{\mathcal{R}}^k \mathbf{e}_{\hat{u}_i}$. To this end, for a trajectory of length l, the computing complexity is reduced from $O(ln_{\mathcal{R}}^2)$ to $O(ln_{\mathcal{R}}(\frac{m_{\mathcal{R}}}{n_{\mathcal{R}}})^k)$, where $\frac{m_{\mathcal{R}}}{n_{\mathcal{R}}}$ denotes the average degree of road graph $G_{\mathcal{R}}$. Both k and the average degree are considered as small constants as the road network is a grid-like sparse graph

TABLE II
COMPLEXITY ANALYSIS OF HMM, SEQ2SEQ-BASED METHODS, AND GRAPHMM (WE OMIT ALL CONSTANT FACTORS)

Methods	#Params	Training	Inference
HMM [6]	$O(1)$	$O(l_{max}S)$	$O(l{n_{\mathcal{R}}}^2)$
Seq2Seq ([3])	$O(n_{\mathcal{R}}d)$	$O(l_{max}Sn_{\mathcal{R}}d)$	$O(ln_{\mathcal{R}}d)$
GraphMM w/o CRF	$O(d^2)$	$O((\frac{m_{\mathcal{R}}+m_{\mathcal{T}}+(n_{\mathcal{R}}+n_{\mathcal{T}})d}{B}+l_{max}n_{\mathcal{R}})Sd)$	$O((\frac{m_{\mathcal{R}}+m_{\mathcal{T}}+(n_{\mathcal{R}}+n_{\mathcal{T}})d}{B}+ln_{\mathcal{R}})d)$
GraphMM	$O(d^2)$	$O((\frac{m_{\mathcal{R}}+m_{\mathcal{T}}+(n_{\mathcal{R}}+n_{\mathcal{T}})d}{B}+l_{max}n_{\mathcal{R}}+l_{max}^3Bd)Sd)$	$O((\frac{m_{\mathcal{R}}+m_{\mathcal{T}}+(n_{\mathcal{R}}+n_{\mathcal{T}})d}{B}+ln_{\mathcal{R}}+l^3Bd)d)$

and are omitted in the asymptotical analysis, which leads to a complexity of $O(ln_{\mathcal{R}})$.

C. Complexity Analysis

We compare **GraphMM** with the classic **HMM** [3] and a line of Seq2Seq-based methods [4], [5], [6]. Since the core components of the Seq2Seq-based methods are identical, we choose **MTrajRec** [4] for comparison. Specifically, we analyze *the number of parameters*, *the training time complexity (per epoch)* and *the inference time complexity (of one input trajectory)* for three types of models (see Table II).

We model the problem input as follows. The road network contains $n_{\mathcal{R}}$ road segments. The training set has totally S trajectories $\mathcal{T}_1, \ldots, \mathcal{T}_S$ of length $l_1, \ldots, l_S$, with $l_{max} = \max\{l_1, \ldots, l_S\}$. Assume that all hidden representations are of dimension d, and all deep models are trained with minibatches of size B. In the inference phase, we consider an input trajectory $\mathcal{T}$ of length l, while the map-matched trajectory has length $O(l)$. For **GraphMM**, the constructed road graph $G_{\mathcal{R}}$ has $n_{\mathcal{R}}$ nodes and $m_{\mathcal{R}}$ edges. Since $G_{\mathcal{R}}$ is a grid-like sparse graph, the average degree $m_{\mathcal{R}}/n_{\mathcal{R}}$ is considered as a constant. The constructed trajectory graph $G_{\mathcal{T}}$ has $n_{\mathcal{T}}$ nodes and $m_{\mathcal{T}}$ edges, satisfying that $n_{\mathcal{T}}, m_{\mathcal{T}} \leq \sum_{i=1}^{S} l_i = O(l_{max}S)$.

For **HMM**, it has only two parameters, the standard deviation of the presumed Gaussian distribution of GPS noise, and the parameter for exponentially distributed transition probability. Each parameter estimation needs a full scan of all S trajectories, resulting in a complexity of $O(l_{max}S)$. For model inference, the Viterbi algorithm incurs $O(ln_{\mathcal{R}}^2)$ complexity for the input trajectory $\mathcal{T}$ of length l. We believe that in the inference phase, the large number of road segments leads to the inefficiency of **HMM**.

For the Seq2Seq-based methods, e.g., **MTrajRec** [4], the RNN-based encoder and decoder contain $O(d^2)$ parameters, while the FC layer and road segment ID embedding layer have $O(n_{\mathcal{R}}d)$ parameters. Since $n_{\mathcal{R}} \gg d$, the number of parameters is asymptotically $O(n_{\mathcal{R}}d)$. In each epoch of the training phase, each step of an RNN incurs $O(d^2)$ time to multiply a d-dimensional vector with $d \times d$-dimensional parameter matrices (we omit constant factors here for multiple gates). Therefore, this step takes $O(l_{max}Sd^2)$ time per epoch. For each trajectory and each step of the decoder, the FC layer takes $O(n_{\mathcal{R}}d)$ time, leading to $O(l_{max}Sn_{\mathcal{R}}d)$ time per epoch. Putting it all together, the training time is $O(l_{max}Sd^2) + O(l_{max}Sn_{\mathcal{R}}d) = O(l_{max}Sn_{\mathcal{R}}d)$ per epoch. (We omit the additional cost of the attention mechanism, which is $O(l_{max}^2Sd^2)$ for the training set.) We can similarly conclude that the inference time for trajectory $\mathcal{T}$ is $O(ln_{\mathcal{R}}d)$.

As for **GraphMM**, it needs $O(d^2)$ parameters for RNN-based encoder and decoder, $O((L+L')d^2)$ parameters for L'-layer road graph convolution and L-layer trajectory graph convolution, and $O(d^2)$ parameters for the conditional layer. Note that L, L' are small constants (we set $L = L' = 2$ in this paper), therefore the number of parameters is asymptotically $O(d^2)$. Therefore, **GraphMM** achieves a balance between **HMM** and **MTrajRec** in terms of model parameters.

Next, we analyze the training efficiency of **GraphMM**. In each epoch, the encoder takes $O(L'm_{\mathcal{R}}d + L'n_{\mathcal{R}}d^2)$ (resp. $O(Lm_{\mathcal{T}}d + Ln_{\mathcal{T}}d^2)$) time for road (resp. trajectory) graph convolution using the full-batch strategy. Note that the first term is the cost of neighborhood aggregation, while the second terms is for feature transformation. As there are S/B minibatches per epoch, the time cost of graph convolution is $O(\frac{S}{B}((m_{\mathcal{R}}+m_{\mathcal{T}})d + (n_{\mathcal{R}}+n_{\mathcal{T}})d^2))$ (we omit the small constants L and L'). The RNN layers in the encoder and the decoder take $O(l_{max}Sd^2)$ time per epoch as Seq2Seq-based methods. For the decoder, hidden similarity computation needs $O(l_{max}n_{\mathcal{R}}d)$ time for every trajectory. Summing it all up, the training time for **GraphMM w/o CRF** is $O(\frac{S}{B}((m_{\mathcal{R}}+m_{\mathcal{T}})d + (n_{\mathcal{R}}+n_{\mathcal{T}})d^2)) + O(l_{max}Sd^2) + O(l_{max}Sn_{\mathcal{R}}d) = O((\frac{m_{\mathcal{R}}+m_{\mathcal{T}}+(n_{\mathcal{R}}+n_{\mathcal{T}})d}{B} + l_{max}n_{\mathcal{R}})Sd)$. For each epoch, with our optimization techniques, the CRF layer has at most $(r+1)l_{max}B$ states. Therefore, the training cost per epoch is bounded by $O(l_{max}((r+1)l_{max}B)^2 \cdot d^2 \cdot S/B) = O(l_{max}^3SBd^2)$. (We omit r, the small constant.) Consequently, the training cost of **GraphMM** is $O((\frac{m_{\mathcal{R}}+m_{\mathcal{T}}+(n_{\mathcal{R}}+n_{\mathcal{T}})d}{B} + l_{max}n_{\mathcal{R}} + l_{max}^3Bd)Sd)$. We can similarly derive the inference time for an input trajectory $\mathcal{T}$. The details are omitted due to space constraints.

Remark: The asymptotical time complexities of **GraphMM** for training and inference are comparable to those of **MTrajRec**, as the terms $\frac{m_{\mathcal{R}}+m_{\mathcal{T}}+(n_{\mathcal{R}}+n_{\mathcal{T}})d}{B}$ and l_{max}^3Bd do not necessarily dominate $l_{max}n_{\mathcal{R}}$. As we will show in the experiments, it turns out that **GraphMM** is about 4 times faster than **MTrajRec**, while **GraphMM w/o CRF** achieves more than one order of magnitude speedup because our parameter size is very much limited.

V. EXPERIMENTAL EVALUATION

A. Experimental Settings

1) Datasets: We use the dataset collected by Tencent Maps for a comprehensive experimental study. We evaluate model accuracy as well as training and inference time. Ablation study and investigation of parameter sensitivity are also conducted.

Tencent: This dataset collects vehicle trajectories for a duration of 3 months, which covers an area of 8.69 km × 7.67 km in the northeast of Beijing. It includes both the road network and a set of vehicle trajectories. The road network contains 8.5 K road segments and 15 K edges between them. Each road segment is associated with the starting and ending GPS location. We totally have 64 K trajectories sampled at a time interval of 15 seconds. Note that the dataset is large in terms of the number of road segments. Meanwhile, the size of available vehicle trajectories is limited, making the problem more difficult. We divide the dataset into training set, validation set and test set with the splitting ratio of 7: 1: 2. We set the *resampling rate* as 50%, 25%, and 12.5% to simulate sparse trajectories with sampling time interval of 30 seconds, 1 minute and 2 minutes, respectively. Each geopoint in the original trajectory is independently sampled. We also use this dataset to evaluate the inductive capability of our model.

2) Baselines: We include the following methods as baselines.

HMM. Adapted from the state-of-the-art method used in Tencent Maps, it is an optimized version of the hidden markov model for efficiency and is associated with several heuristic refinements for accuracy. In particular, the emission probabilities are computed by considering 1)the distance from the GPS location to the road segment and 2)the consistency of travel direction shown by the trajectory and road segment direction. For the computation of transition probabilities, we eliminate most unrelated road segments and keep a small candidate set for each step of prediction, which greatly improves model efficiency. For each trajectory point, we define its candidate road segments as those of which the minimal distances to the current and five previous trajectory points is less than 200 meters.

MTrajRec and MTrajRec w/o CM [4]: It is the state-of-the-art method for vehicular map matching. We adopt the following modifications for our problem. As it is impossible to get the moving ratio for the aforementioned real-world datasets, we deactivate the multi-task learning module. Similarly, we do not use any external attributes such as weather conditions and POI distributions. However, the time information (i.e., timestamp for each trajectory point) is preserved. Besides, **MTrajRec w/o CM** denotes the variant without the constraint mask layer, which contributes most to model accuracy as shown in [4].

Note that we do not include DeepMM [5], [6], DHTR [7] and other related approaches (e.g., [26]) for comparison for the following reasons. First, as shown in [4], **MTrajRec** outperforms most methods [7], [26]. Second, the main component of [5] and [7] are identical to **MTrajRec w/o CM**. Third, we cannot get the implementation of [5] and [7] from the authors.

3) Implementation: We implement **GraphMM** in PyTorch and adopt PyTorch Geometric, and get the code of **MTrajRec** from the authors. By default we fix the grid length as 50 m as suggested by [4]. We adapt the same parameter configuration as described in the source code of **MTrajRec**, except that 1)the learning rate is modified to 0.0005 and 2)we train the model for 100 epochs for better accuracy. (The model does not converge for 20 epoches.) For **GraphMM**, we fix the learning rate as 0.0001 and the hidden dimensions as 256. To speed up the CRF layer, we set $r = 3$ to approximate $Z(\tilde{\mathbf{Y}})$ in model training and set $r = 5$ during inference, and set $k = 4$ for pairwise potential (13). We train our models for 200 epochs. All experiments are conducted on a Ubuntu server with an NVIDIA GeForce RTX 3090 Ti GPU of 24 GB memory. Our source code and data have been made available at https://github.com/GraphMMmaster/GraphMM-Master.

4) Evaluation Metrics: Let $Ts = \{\mathcal{T}_1, \ldots, \mathcal{T}_{S'}\}$ denote the test set, which contains S' trajectories. For each trajectory $\mathcal{T}_i = (p_{i1}, \ldots, p_{il_i})$, let l_i denote its length, i.e., the number of trajectory points. Motivated by real-world applications in Tencent Maps, we adopt the following two metrics for model evaluation.

Trajectory-level Accuracy: This is the evaluation metric used by Tencent Maps. For each trajectory $\mathcal{T}_i$, let $(u_{i1}, \ldots, u_{il_i})$ be the ground truth route, and $(\hat{u}_{i1}, \ldots, \hat{u}_{il_i})$ the predicted road segments. The prediction accuracy for $\mathcal{T}_i$ is defined as

$$\text{Accuracy for } \mathcal{T}_i = \frac{\sum_{j=1}^{l_i} \mathbb{I}[u_{ij} = \hat{u}_{ij}]}{l_i}, \tag{15}$$

where $\mathbb{I}[A]$ is the indicator variable, which takes value 1 if condition A is satisfied, and takes value 0 otherwise. The trajectory-level accuracy is then defined as the average of prediction accuracy over all trajectories in Ts (referred to as Accuracy (T)):

$$\text{Accuracy (T) for } Ts = \frac{\sum_{i=1}^{S'} \text{Accuracy for } \mathcal{T}_i}{S'}. \tag{16}$$

Ratio of Longest Common Subsequence (R-LCS): We also adopt longest common subsequence (LCS) to evaluate the similarity of two sequences of same length. In particular, it is defined as

$$\text{R-LCS for } \mathcal{T}_i = \frac{\text{LCS}\{(u_{i1}, \ldots, u_{il_i}), (\hat{u}_{i1}, \ldots, \hat{u}_{il_i})\}}{l_i}, \tag{17}$$

$$\text{R-LCS for } Ts = \frac{\sum_{i=1}^{S'} \text{R-LCS for } \mathcal{T}_i}{S'}. \tag{18}$$

Note that we are the first to adopt prediction accuracy for evaluation, while both metrics take the ground truth and the predicted routes as *sequences*. This is because *the order of predicted road segments is of vital importance for real-world applications*. As a consequence, we will not focus on the evaluation metrics for sets such as precision and recall. Although the distance between a GPS location and its matched coordinates is an important measure, we do not adopt it because 1)for real-world application data such as Tencent, this information is unavailable, and 2) the reported GPS location might have large deviation from the exact position of the vehicle, and distance-based metrics are not good choices.

B. Evaluation of Prediction Accuracy

We evaluate the prediction accuracy of **HMM**, **MTrajRec**, and **GraphMM** on the Tencent dataset, as illustrated in Table III. For the transductive setting, we use the bold font to highlight the best result, and use underline for the second best values. Surprisingly, **HMM** is quite robust with different keep ratio, probably due to various optimizations adopted in the industrial implementation. Note that our version of HMM

TABLE III
EVALUATION OF PREDICTION ACCURACY ON THE TENCENT DATASET UNDER TRANSDUCTIVE AND INDUCTIVE SETTINGS

Methods	50% (30 seconds)		25% (1 minute)		12.5% (2 minutes)	
	Accuracy (T)	R-LCS	Accuracy (T)	R-LCS	Accuracy (T)	R-LCS
HMM	0.3655	0.4162	0.3443	0.3790	0.3519	0.3726
MTrajRec w/o CM	0.5589	0.638	0.4193	0.554	0.2982	0.4724
MTrajRec	0.6154	0.6752	0.4992	<u>0.5997</u>	0.3706	0.5071
GraphMM w/o CRF	<u>0.657</u>	<u>0.704</u>	<u>0.5224</u>	0.5995	<u>0.4171</u>	<u>0.5155</u>
GraphMM	**0.6713**	**0.7174**	**0.5336**	**0.6196**	**0.4251**	**0.5395**
GraphMM w/o CRF (inductive)	0.6428	0.6923	0.5114	0.605	0.4154	0.5141
GraphMM (inductive)	**0.6622**	**0.7113**	**0.5271**	**0.6154**	**0.42**	**0.5342**

directly makes prediction on the sparse trajectory rather than first recovering a high-sampling-rate one. However, it is still surpassed by the deep learning models as they are able to extract complicated spatiotemporal patterns from the training data. On the other hand, for **MTrajRec**, the distance-based mask layer has a large impact on the performance. Nonetheless, even without the CRF layer, our approach (**GraphMM w/o CRF**) consistently achieves better accuracy than **MTrajRec** in terms of Accuracy(T) (by 6.76%, 4.65%, and 12.55% respectively) and has better or comparable performance on R-LCS, as our model is able to utilize more types of correlations between trajectories and roads. For **GraphMM**, it has the best performance over all settings. It outperforms **MTrajRec** by 9.08%, 6.89%, and 14.71% in terms of Accuracy(T) and by 6.25%, 3.32%, and 6.39% for R-LCS, for resampling rate of 50%, 25%, and 12.5% respectively. Notably, all methods achieve better results on R-LCS, because it is more tolerable to position-wise error. As the resampling rate decreases, the performance of all deep models degrade significantly. We believe that this phenomenon can be alleviated with more sophisticated representation learning methods for sparse trajectories, such as more powerful GNNs for trajectory graph convolution. We also note that there exists a large room for accuracy improvement by exploiting the correlation of road segments, e.g., by adding more features to the input of the conditional layer. This observation is derived from the fact that our method manages to identify the ground truth as one of the top candidates but fails to distinguish it from others. We leave it as future work.

Evaluation of Inductive Capability: Our experimental setting for inductive map matching follows that of GNN [11], [12]. For our baselines, note that **HMM** can be directly applied to this setting because it does not have the training phase, and consequently it has the same prediction accuracy. However, **MTrajRec** and other existing Seq2Seq-based methods cannot be adapted to this scenario (see Section III-C1). To be precise, we only use trajectories of the training set to construct the trajectory graph, and train the model on a subset of all road segments which covers all training data. In the model inference phase, remaining road segments are added to the road network, and we update the trajectory graph accordingly with the test set. However, the model is not retrained but directly used for prediction.

We show the experimental results of our models in the bottom of Table III. Interestingly, our models have nearly no degeneration of prediction accuracy, and significantly surpass all baselines conducted in transductive setting. Probably this is because the training data already contains important patterns of roads and trajectories as well as the correlations.

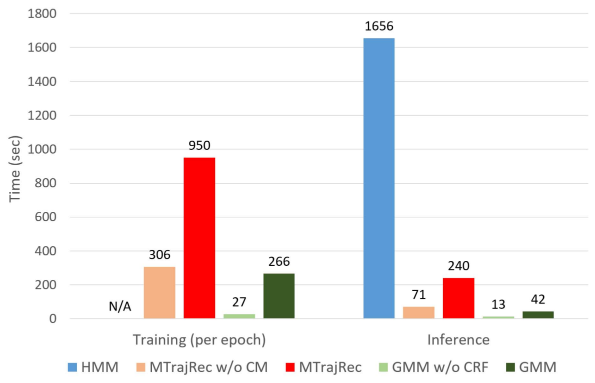


Fig. 4. Training time per epoch and inference time on the test set, with a resampling rate of 50%.

C. Evaluation of Model Efficiency

Fig. 4 demonstrates the efficiency of different methods with a resampling rate of 50%. We omit other settings of resampling rate because they demonstrate similar conclusions. We report the training time *per epoch* and the inference time on the test set with about 12 K trajectories.

HMM does not have model training time as it is adapted from industrial systems. Unfortunately, it incurs the longest inference time even with various optimizations. This is in consistent with previous literature and coincides with our theoretical analysis. **MTrajRec** and its variant take much time for training because they contain tremendous numbers of parameters. Actually, the model training of **MTrajRec** (with 100 epochs) takes more than two days. With the same number of epoches, it converges slower than our models. In contrast, **GraphMM w/o CRF** is more than 10× (resp. 35×) faster than **MTrajRec w/o CM** (resp. **MTrajRec**) for model training, and is about 5× and 18× faster for model inference, respectively. Even with the CRF layer, our model is significantly faster than **MTrajRec w/o CM**. It also has 4 × speedup for model training and 8 × speedup for inference compared to **MTrajRec**. This is because our model has much fewer parameters as we employ two graph representation layers for the encoder along with the hidden similarity computation layer for the decoder. Besides, our proposed acceleration techniques are proved effective for the CRF layer with over 8 K labels.

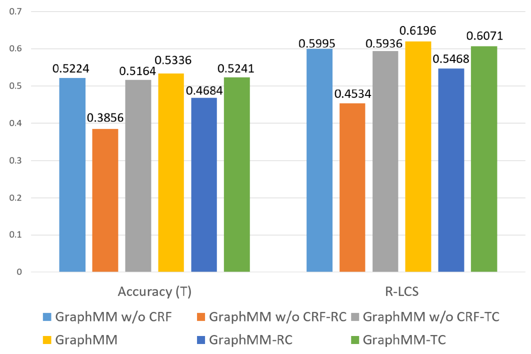


Fig. 5. Ablation study of our models (with grid size of 50 meters and a resampling rate of 25%).

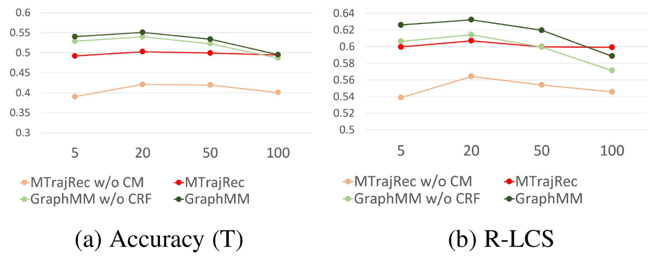


Fig. 6. Prediction accuracy varying grid size (resampling rate = 25%).

D. Ablation Study

To validate the effectiveness of different modules, we replace the road graph convolution layer (denoted as "-RC") and the trajectory graph convolution layer (denoted as "-TC") as an MLP with same hidden dimension and number of layers, respectively. Note that we cannot fully remove these two modules nor the hidden similarity computation layer. Otherwise, our model degenerates to a Seq2Seq model. Notably, the road graph convolution and the hidden similarity computation layers are inseparable as the latter relies on the former to compute road representations. Without modeling trajectory-road correlation, the model becomes a variant of [4].

It can be concluded from Fig. 5 that road graph convolution plays an important role in model accuracy. On the other hand, the contribution of the trajectory graph convolution layer is not that significant. We believe that a major reason is the limited power of GCN. Also note that our model totally ignores the timestamp of each geopoint and only focuses on the spatial information, thus does not utilize temporal patterns in the trajectories as opposed to most existing studies. We leave the application of more powerful GNNs (e.g., temporal GNNs) as future work.

E. Parameter Sensitivity

We conduct parameter sensitivity study for grid size, which is believed to have non-negligible impact on model performance. As Fig. 6 shows, reducing grid size from the default value (50 meters) generally enhances the accuracy of both **GraphMM** and **MTrajRec** as well as their gap. Meanwhile, enlarging grid size deteriorates the accuracy of both models but has greater affect on **GraphMM**. This is reasonable as both graph convolution layers of **GraphMM** depend on the grids, and a fine-grained partition of the map keeps more geographical information.

VI. Related Work

A. State-of-The-Art Methods for Map Matching

Existing studies of the vehicular map matching problem can be roughly categorized as the traditional hidden Markov model-based method [3] and the recent methods [4], [5], [7] that employ the Seq2Seq model [27] with various model improvements.

1) HMM: It treats sparse trajectories as observations and road segments as states. Specifically, given an input trajectory $\mathcal{T} = (p_1, \dots, p_l)$ of length l, HMM predicts an l-length sequence of road segments $\mathcal{R} = (r_1, \dots, r_l)$, by maximizing the joint probability over $\mathcal{T}$ and $\mathcal{R}$:

$$P(\mathcal{T}, \mathcal{R}) = \pi(r_1) \prod_{i=1}^{l} P(p_i|r_i) \prod_{i=1}^{l-1} P(r_{i+1}|r_i), \tag{19}$$

where the initial state probabilities $\pi(r_1)$ follow the uniform distribution over all road segments, whereas the emission probabilities $P(p_i|r_i)$ and transition probabilities $P(r_{i+1}|r_i)$ are empirically set to follow some predefined probability distributions of estimated driving distance, with only a few learnable parameters. Given a sparse trajectory, the Viterbi algorithm is adopted to recover the matched road segments. Although HMM is effective for high-sampling-rate trajectories, it is incapable of learning complicated spatial or temporal patterns from the trajectory data, which leads to low accuracy when dealing with sparse trajectories.

2) Seq2Seq-Based Methods: Nearly all Seq2Seq-based methods [4], [5], [6], [7] apply the following encoder-decoder paradigm as the main component of the model:

$$\mathbf{h}_{\mathcal{T}} = \mathbf{RNNEncoder}(\mathcal{T}), \tag{20}$$

$$\mathcal{R} = (\mathbf{Attentional})\mathbf{RNNDecoder}(\mathbf{h}_{\mathcal{T}}). \tag{21}$$

First, the RNN-based encoder transforms the input trajectory $\mathcal{T}$ into $\mathbf{h}_{\mathcal{T}}$, the hidden representation of the trajectory. Next, the RNN-based decoder recovers the map-matched trajectory $\mathcal{R}$ from $\mathbf{h}_{\mathcal{T}}$. If the attention mechanism is employed, hidden information at every step of the encoder will be explicitly exploited in the decoding stage. As a consequence, for most Seq2Seq-based models, their main components have almost identical expressive capability for map matching. Among them, **MTrajRec** [4] is the state-of-the-art method for map matching of sparse vehicle trajectories.

MTrajRec [4]: It follows the encoder-decoder paradigm, where the encoder is based on GRU and includes an attribute module by considering additional environmental and road features, and the RNN-based decoder is carefully designed to integrate the standard attention mechanism, the constraint mask layer, and the multi-task learning module to improve accuracy. Particularly, the most powerful component is the *constraint mask layer*, which prunes impossible trajectory-road matchings according to their Euclidean distance. *The intuition is that a trajectory point should not be matched to road segments too* *far away*. To be precise, at the i-th step of the decoder, the hidden representation $\mathbf{h}_i \in \mathbb{R}^d$ is multiplied by a fully connected (FC) layer with learnable matrix $\mathbf{W}_{\text{FC}} \in \mathbb{R}^{d \times n_{\mathcal{R}}}$ and then passed through the softmax function to derive the probability vector $\hat{\mathbf{y}}_i \in \mathbb{R}^{n_{\mathcal{R}}}$:

$$P(\hat{u}_i|\mathbf{h}_i) = \hat{y}_{i\hat{u}_i} = \frac{\exp(\mathbf{h}_i^\intercal \mathbf{W}_{\text{FC}} \mathbf{e}_{\hat{u}_i}) \odot c_{i\hat{u}_i}}{\sum_{j \in [1, n_{\mathcal{R}}]} \exp(\mathbf{h}_i^\intercal \mathbf{W}_{\text{FC}} \mathbf{e}_j) \odot c_{ij}}. \quad (22)$$

Note that $\hat{y}_{i\hat{u}_i}$, the $\hat{u}_i$-th component of $\hat{\mathbf{y}}_i$, indicates the probability that road segment $\hat{u}_i$ is matched, while $\mathbf{e}_i$ is the indicator vector with i-th component set to 1 and all others set to 0. **MTrajRec** uses a mask $\mathbf{c}_i \in \mathbb{R}^{n_{\mathcal{R}}}$ to filter out road segments of distance larger than a predefined threshold τ. Specifically, the j-th component of $\mathbf{c}_i$ is set to $c_{ij} = \exp(-\frac{d_{ij}^2}{\beta^2})$, where d_{ij} is the Euclidean distance between the i-th trajectory point and the j-th road segment. Moreover, if the distance is larger than τ, c_{ij} is set to 0. In [4], τ and β are set to 50 (in meters) and 15, respectively.

Along with other previous work, **MTrajRec** formulates the map matching problem as the multi-label classification problem, by taking the set of all road segments as labels. Thus, it is natural to adopt the cross-entropy loss:

$$\mathcal{L}_{CE} = - \sum_{(\mathcal{T},\mathcal{R}) \in Tr} \sum_{i=1}^{|\mathcal{R}|} \sum_{k=1}^{n_{\mathcal{R}}} y_{ik} \log \hat{y}_{ik}, \quad (23)$$

where Tr denotes the training set. Each training data instance is a pair of sparse trajectory $\mathcal{T}$ and matched road sequence $\mathcal{R}$, whereas $\mathbf{y}_i = (y_{i1}, \ldots, y_{in_{\mathcal{R}}})$ is a one-hot vector indicating the ground truth for each index i of the matched road sequence. For other representative Seq2Seq-based models, DeepMM [5], [6] leverages BiLSTM as the encoder and LSTM as the decoder, and incorporates the standard attention mechanism [28]. Another refinement step based on heuristic strategies for the topology continuity of trajectory is employed as postprocessing. In addition, DeepMM studies the techniques for trajectory augmentation, which is considered as an orthogonal contribution to model design. DHTR [7] adopts a similar model framework to DeepMM. Besides, it improves the decoder by considering region constraint by utilizing the information of current input trajectory. It also incorporates a Kalman Filter component with the Seq2Seq model to calibrate estimation. The model does not make use of the road network or any other graph-based information.

Remark: As for the modeling of correlations proposed in our paper, note that these models [4], [5], [6], [7] explicitly capture intra-trajectory correlation by the adopted sequential models, of which the expressive power is guaranteed by that of RNN [24]. In contrast, inter-trajectory correlation can only be implicitly learned by feeding all trajectories to the deep model. By adding various components and mechanisms other correlations might be implicitly considered. For example, the road segment ID embedding layer along with teacher forcing partially captures the correlation between road segments. However, it is still limited by the sequential nature of RNNs. It seems that existing approaches cannot leverage trajectory-road correlation as they cannot learn effective representation of road segments.

Another interesting observation is that existing deep models [4], [5], [6], [7] lack inductive capability, say, training on one region of the map and conducting inference on another region. Note that even for the same region but with one road segment added, these models have to be retrained from scratch. This is because the modification of the label set results in the change of model structure, for example, the dimension of the FC layer needs to be updated.

B. Other Related Work

Other Variants of Map Matching: A line of works [29], [30] investigate *pedestrian map matching*, a more difficult task as pedestrians can be either indoors or outdoors, while road information might be unavailable. *Online map matching* [31], [32], [33], [34] aims to predict the partial route by collecting the real-time trajectory and poses greater challenges in terms of prediction accuracy and response time.

Other Related Problems: A similar problem to map matching is the *next-step location prediction* [26], [35], [36], which predicts the next few locations based on the travel history. Recently, a few studies [7], [8] focus on *trajectory recovery*, which recovers the high-sampling-rate trajectory from a low-sampling-rate one. Another line of research [37], [38], [39] studies travel time prediction, and the main idea is to capture spatiotemporal dependencies.

Existing Studies Sharing Similar Ideas: We observe that a few recent studies adopt similar ideas for model design. For example, TrajNet [40] and Trajectory WaveNet [25] tackle the traffic forecasting problem which predicts the traffic speeds in the near future given historical data and the trajectories. For the first time, they treat vehicle trajectories as first-class citizens. In particular, these models first extract temporal patterns of the road segments and then pass them to a sequential model in which feature propagation along a trajectory is conducted. This shares some resemblance with our graph-augmented trajectory encoder. From the perspective of adopted deep neural networks, except for the Seq2Seq models, other techniques have been employed such as the attentional neural network [8]. We note that a few work has adopted graph learning techniques for road segment or trajectory representation [36], [41], [42]. However, they focus on other problem settings and their models cannot be directly applied to our problem.

VII. Conclusion

We propose **GraphMM**, a graph-based approach for vehicular map matching. Apart from leveraging sequential models to extract intra-trajectory correlation, our model improves matching accuracy by integrating inter-trajectory correlation via trajectory graph convolution and trajectory-road correlation via road graph convolution and hidden similarity computation. Additionally, we enhance the classic sequential decoder with a graph-based conditional model to incorporate correlation between road segments. The necessity of capturing these correlations and the expressive capability of our proposed model are formally proved. **GraphMM** is the first map matching model that explicitly considers various data and label correlations via graph modeling, and the first to provide inductive capability. Experimental results show that **GraphMM** outperforms both industrial implementation of the hidden Markov model as well as state-of-the-art Seq2Seq-based methods in terms of prediction accuracy, while enhances training and inference efficiency by up to an order of magnitude. In the future, we plan to extend our model to larger problem sizes and the online problem setting. From the technical point of view, we will conduct more extensive investigation on the correlation of road segments to further improve model accuracy.

References

[1] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *Proc. IEEE 11th Int. Conf. Mobile Data Manage.*, 2010, pp. 43–52.

[2] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Proc. IEEE 28th Int. Conf. Data Eng.*, 2012, pp. 1144–1155.

[3] P. Newson and J. Krumm, "Hidden Markov map matching through noise and sparseness," in *Proc. 17th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2009, pp. 336–343.

[4] H. Ren et al., "MTrajRec: Map-constrained trajectory recovery via seq2seq multi-task learning," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 1410–1419.

[5] J. Feng et al., "DeepMM: Deep learning based map matching with data augmentation," *IEEE Trans. Mobile Comput.*, vol. 21, no. 7, pp. 2372–2384, Jul. 2020.

[6] K. Zhao et al., "DeepMM: Deep learning based map matching with data augmentation," in *Proc. 27th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2019, pp. 452–455.

[7] J. Wang, N. Wu, X. Lu, W. X. Zhao, and K. Feng, "Deep trajectory recovery with fine-grained calibration using kalman filter," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 921–934, Mar. 2021.

[8] T. Xia et al., "AttnMove: History enhanced trajectory recovery via attentional network," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 4494–4502.

[9] Q. Huang, H. He, A. Singh, S.-N. Lim, and A. R. Benson, "Combining label propagation and simple models out-performs graph neural networks," 2020, arXiv:*2010.13993*.

[10] H. Wang and J. Leskovec, "Unifying graph convolutional neural networks and label propagation," 2020, arXiv:*2002.06755*.

[11] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[12] W. L. Hamilton, "Graph representation learning," *Synth. Lectures Artifical Intell. Mach. Learn.*, vol. 14, no. 3, pp. 1–159, 2020.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, arXiv:*1609.02907*.

[14] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2018, arXiv:*1810.00826*.

[15] J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang, "How powerful are K-hop message passing graph neural networks," 2022, arXiv:*2205.13328*.

[16] X. Wang and M. Zhang, "How powerful are spectral graph neural networks," 2022, arXiv:*2205.11172*.

[17] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1725–1735.

[18] G. Li, C. Xiong, A. Thabet, and B. Ghanem, "DeeperGCN: All you need to train deeper GCNs," 2020, arXiv:*2006.07739*.

[19] M. Chen et al., "Scalable graph neural networks via bidirectional propagation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 14556–14566.

[20] R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. S. Bhowmick, "Scaling attributed network embedding to massive graphs," 2020, arXiv:*2009.00826*.

[21] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI, Ser.*, vol. 2, no. 9, pp. 12–16, 1968.

[22] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture," in *Proc. IEEE Intell. Veh. Symp.*, 2018, pp. 1672–1678.

[23] H. Wang, R. Hu, Y. Zhang, L. Qin, W. Wang, and W. Zhang, "Neural subgraph counting with Wasserstein estimator," in *Proc. Int. Conf. Manage. Data*, 2022, pp. 160–175.

[24] V. Khrulkov, A. Novikov, and I. Oseledets, "Expressive power of recurrent neural networks," 2017, arXiv:*1711.00811*.

[25] B. Hui, D. Yan, H. Chen, and W.-S. Ku, "Trajectory waveNet: A trajectory-based model for traffic forecasting," in *Proc. IEEE Int. Conf. Data Mining*, 2021, pp. 1114–1119.

[26] J. Feng et al., "DeepMove: Predicting human mobility with attentional recurrent networks," in *Proc. World Wide Web Conf.*, 2018, pp. 1459–1468.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, arXiv:*1409.0473*.

[29] M. Ren, "Advanced map matching technologies and techniques for pedestrian/wheelchair navigation," PhD thesis, Sch. Inform. Sci., Univ. Pittsburgh, Pittsburgh, PA, USA,, 2012.

[30] L. Zhang, M. Cheng, Z. Xiao, L. Zhou, and J. Zhou, "Adaptable map matching using PF-net for pedestrian indoor localization," *IEEE Commun. Lett.*, vol. 24, no. 7, pp. 1437–1440, Jul. 2020.

[31] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden markov model for real-time traffic sensing applications," in *Proc. IEEE 15th Int. Conf. Intell. Transp. Syst.*, 2012, pp. 776–781.

[32] M. Hashemi and H. A. Karimi, "A critical review of real-time map-matching algorithms: Current issues and future directions," *Comput. Environ. Urban Syst.*, vol. 48, pp. 153–165, 2014.

[33] T. Hunter, P. Abbeel, and A. Bayen, "The path inference filter: Model-based low-latency map matching of probe vehicle data," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 507–529, Apr. 2014.

[34] S. Taguchi, S. Koide, and T. Yoshimura, "Online map matching with route prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 338–347, Jan. 2019.

[35] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 194–200.

[36] K. Liu et al., "Modeling trajectories with multi-task learning," in *Proc. IEEE 23rd Int. Conf. Mobile Data Manage.*, 2022, pp. 208–213.

[37] A. Derrow-Pinion et al., "ETA prediction with graph neural networks in Google maps," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 3767–3776.

[38] Z. Zhang, H. Wang, Z. Fan, J. Chen, X. Song, and R. Shibasaki, "Route to time and time to route: Travel time estimation from sparse trajectories," 2022, arXiv:*2206.10418*.

[39] Z. Chen et al., "Interpreting trajectories from multiple views: A hierarchical self-attention network for estimating the time of arrival," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 2771–2779.

[40] B. Hui, D. Yan, H. Chen, and W.-S. Ku, "TrajNet: A trajectory-based deep learning model for traffic prediction," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 716–724.

[41] H. Sun, C. Yang, L. Deng, F. Zhou, F. Huang, and K. Zheng, "PeriodicMove: Shift-aware human mobility recovery with graph neural network," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 1734–1743.

[42] D. Yao, H. Hu, L. Du, G. Cong, S. Han, and J. Bi, "TrajGAT: A graph-based long-term dependency modeling approach for trajectory similarity computation," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 2275–2285.

Yu Liu received the BS degree in computer science from Shandong University, in 2011, and the MEng and PhD degrees from the School of Information, Renmin University, in 2014 and 2018, respectively. He is now a lecturer with Beijing Jiaotong University. He has published more than 10 refereed papers in various journals and conferences. His current research interests include scalable graph algorithms and graph learning methods.

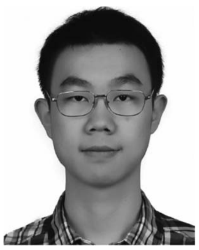
Qian Ge received the BS degree in computer science from Peking University, in 2021. He is currently working toward the master's degree with the Academy for Advanced Interdisciplinary Studies, Peking University. His research interests include approximate graph algorithms and graph neural networks.

Wei Luo received the BS degree in computer science from Southwest Jiaotong University. He is currently working toward the MEng degree with the School of Computer and Information Technology, Beijing Jiaotong University. His research interests include graph neural networks and graph learning approaches for NP-hard problems.

Qiang Huang received the PhD degree in operation research from the Institute of Applied Mathematics, Academy of Mathematics & Systems Science, Chinese Academy of Sciences in 2014. He is now working with Tencent Inc., Beijing, China. His research interests include undirected graphical models and graph neural networks.

Lei Zou received the BS and PhD degrees in computer science from the Huazhong University of Science and Technology (HUST), in 2003 and 2009, respectively. Now, he is a professor with the Wangxuan Institute of Computer Technology, Peking University. His research interests include graph database and semantic data management.

Haixu Wang received the BS degree in computer science from Beijing Jiaotong University, in 2021. He is currently working toward the MEng degree with the School of Computer and Information Technology, Beijing Jiaotong University. His research interests include applications of graph neural networks and their efficiency.



Xin Li received the BS and MEng degree from the School of Information, Renmin University of China, in 2011 and 2014, respectively. He is now working with Tencent Inc., Beijing, China. His current research interests include spatial database and geo-positioning.

Chang Liu received the bachelors and master's degrees from the Beihang University of China, in 2007 and 2010, respectively, and is currently working with Tencent as an expert engineer. His current research interests include spatial database and geo-positioning.