DiffMM: Efficient Method for Accurate Noisy and Sparse Trajectory Map Matching via One Step Diffusion

Chenxu Han[1], **Sean Bin Yang**[2], **Jilin Hu**[1,3*]

[1]School of Data Science and Engineering, East China Normal University, Shanghai, China
[2]Department of Computer Science, Aalborg University, Aalborg, Denmark
[3]KLATASDS-MOE, East China Normal University, Shanghai, China
cxhan@stu.ecnu.edu.cn, seany@cs.aau.dk, jlhu@dase.ecnu.edu.cn

Abstract

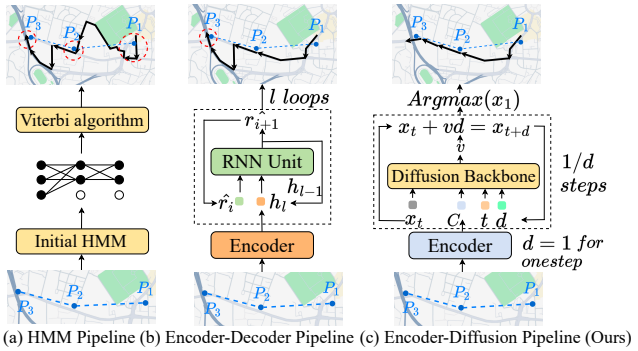
Map matching for sparse trajectories is a fundamental problem for many trajectory-based applications, e.g., traffic scheduling and traffic flow analysis. Existing methods for map matching are generally based on Hidden Markov Model (HMM) or encoder-decoder framework. However, these methods continue to face significant challenges when handling noisy or sparsely sampled GPS trajectories. To address these limitations, we propose *DiffMM*, an encoder–diffusion-based map matching framework that produces effective yet efficient matching results through a one-step diffusion process. We first introduce a road segment-aware trajectory encoder that jointly embeds the input trajectory and its surrounding candidate road segments into a shared latent space through an attention mechanism. Next, we propose a one step diffusion method to realize map matching through a shortcut model by leveraging the joint embedding of the trajectory and candidate road segments as conditioning context. We conduct extensive experiments on large-scale trajectory datasets, demonstrating that our approach consistently outperforms state-of-the-art map matching methods in terms of both accuracy and efficiency, particularly for sparse trajectories and complex road network topologies.

Code — https://github.com/decisionintelligence/DiffMM

Introduction

Map matching is a fundamental component in map service that aligns the vehicle or human trajectory records with the underlying the road network. It is of vital importance in many trajectory-based applications, including vehicle navigation (Jain et al. 2021), traffic flow prediction (Qiu et al. 2024, 2025), traffic scheduling (Stenneth et al. 2011), route optimization (Yuan et al. 2013). For instance, in services like Google Maps, accurate map matching is crucial for delivering reliable navigation by precisely localizing users and estimating real-time traffic conditions on the road network.

Map matching has been extensively studied over the past years, resulting in a wide range of proposed methods. These approaches can generally be categorized into two main groups: non-learning-based methods (Newson and Krumm 2009; Huang et al. 2018; Gong et al. 2018; Mohamed, Aly, and Youssef 2017) and learning-based methods (Feng et al. 2022; Liu et al. 2024; Chen et al. 2023). One of the most widely used non-learning-based approaches for map matching is the Hidden Markov Model (HMM) (Newson and Krumm 2009), which models the sequence of observed GPS points as emissions from a hidden sequence of road segments, effectively capturing both spatial proximity and temporal continuity. In contrast, learning-based methods have gained increasing attention in recent years. Several end-to-end approaches (Feng et al. 2022; Liu et al. 2024; Chen et al. 2023) have been proposed, which consistently outperform traditional HMM-based methods. These approaches are primarily based on the sequence-to-sequence (Seq2Seq) paradigm and adopt an encoder-decoder framework to model the mapping from noisy GPS trajectories to the corresponding road segment sequences.



(a) HMM Pipeline (b) Encoder-Decoder Pipeline (c) Encoder-Diffusion Pipeline (Ours)

Figure 1: Comparison of Different Map Matching Pipelines. The black arrows represent the result matched road segments and the red dashed circles highlight the wrong result.

Taking Figure 1 as as example, we observe the following: (1) As shown in Figure 1(a), traditional HMM-based methods are highly sensitive to noisy and sparsely sampled trajectory data, leading to degraded matching performance, particularly in the regions highlighted by the red dashed circle; (2) As shown in Figure 1(b), although learning-based methods demonstrate improved performance over non-learning-based approaches, their matching accuracy is still adversely affected by sparse trajectory data. Moreover, the autoregres-

*Corresponding Author

sive nature of RNN-based decoders introduces inefficiencies and tends to accumulate errors during the decoding process, ultimately leading to suboptimal results. To this end, despite substantial advancements, both non-learning-based and learning-based map matching methods continue to face significant challenges when handling noisy or sparsely sampled GPS trajectories.

Specifically, we highlight the following two major limitations of existing methods:

- **Sensitive to noise in trajectory data.** Due to environmental complexity and limitations of positioning devices, GPS data often exhibit significant drift, causing recorded points to deviate from their true locations. Such noise severely impacts the accuracy of map matching, particularly for distance-based methods such as HMM, which rely heavily on spatial proximity for candidate selection and state transitions.
- **Degraded performance on sparse trajectories.** To reduce storage costs, many map services adopt low-frequency sampling strategies, resulting in sparsely recorded GPS trajectories. While this approach is storage-efficient, it poses significant challenges for map matching. The limited number of observations hampers the ability to capture temporal and spatial continuity between consecutive points, making it difficult to infer accurate road segment sequences. Consequently, the matching accuracy is often significantly degraded when dealing with sparse trajectories.

To address these limitations, we propose *DiffMM*, an encoder–diffusion-based map matching framework that produces effective yet efficient matching results through a one-step diffusion process. Specifically, we first introduce a road segment-aware trajectory encoder that jointly embeds the input trajectory and its surrounding candidate road segments into a shared latent space. This is achieved through an attention mechanism that captures the interactions between trajectory points and road segments, allowing the model to focus on spatially and contextually relevant features. To construct the candidate set, we adopt a δ-meter radius strategy, which selects all road segments within a predefined distance δ from each GPS point, ensuring that the candidate segments are both spatially meaningful and computationally tractable. Next, we propose a distribution-based map matching model that leverages the joint embedding of the trajectory and candidate road segments as conditioning context. Based on this representation, we formulate the map matching task as a one-step diffusion process, where the target road segment distribution is recovered via a denoising mechanism applied to Gaussian noise. Specifically, we adopt a shortcut model to approximate the conditional distribution in a single denoising step, significantly enhancing the efficiency of both training and inference. We conduct extensive experiments on large-scale trajectory datasets to evaluate the effectiveness of *DiffMM*. The results demonstrate that our approach consistently outperforms state-of-the-art map matching methods in terms of both accuracy and efficiency, particularly under challenging conditions involving highly sparse trajectories and complex road network topologies. We summarize our contributions as follows:

- We propose a novel one-step diffusion-based map matching framework, *DiffMM*. To the best of our knowledge, we are the first to model Map Matching through the conditional distribution, which is within diffusion paradigm and allows us to leverage the information of trajectory and road network.
- We propose a road segment-aware trajectory encoder that jointly embeds the input trajectory and its surrounding candidate road segments into a shared latent space through an attention mechanism.
- We propose a one step diffusion method to realize map matching through a shortcut model by leveraging the joint embedding of the trajectory and candidate road segments as conditioning context.
- We conduct extensive experiments on large-scale trajectory datasets, demonstrating that our approach consistently outperforms state-of-the-art map matching methods in terms of both accuracy and efficiency, particularly for sparse trajectories and complex road network topologies.

Related Work

Map Matching

There is a plethora of studies focus on map matching. The classic study (Newson and Krumm 2009) leverages Hidden Markov Model to find the most likely matched road segments, and it have many variants (Huang et al. 2018; Gong et al. 2018; Mohamed, Aly, and Youssef 2017; Yang and Gidofalvi 2018). (Zheng et al. 2012) present a history-based route infer system. (Lou et al. 2009) leverage the road network and the temporal features of trajectories to construct a candidate graph. FMM (Yang and Gidofalvi 2018) apply some acceleration mechanisms on HMM. Recent trend is to extract patterns from historical trajectories for map matching. DeepMM (Feng et al. 2022) is an end-to-end deep learning method with data augmentation. GraphMM (Liu et al. 2024) is a graph-based method which incorporate graph neural networks to leverage both intra-trajectory and inter-trajectory correlation for map matching. RNTrajRec (Chen et al. 2023) is an approach to solve both map matching and trajectory recovery by considering the road network structure. Different from the existing methods, we formulate map matching as a problem of learning a conditional distribution and solve it by DiffMM.

Diffusion Models

Diffusion models (Ho, Jain, and Abbeel 2020; Sohl-Dickstein et al. 2015; Song, Meng, and Ermon 2020; Frans et al. 2024) are a class of generative models. There are rich studies on diffusion models in various domains, and the diffusion models have been widely used for many applications due to their powerful generative capabilities. These applications include image generation (Austin et al. 2021; Dhariwal and Nichol 2021; Rombach et al. 2022; Sinha et al. 2021), time series prediction and imputation (Rasul et al. 2021; Tashiro et al. 2021), text generation (Gong et al. 2022; Li

et al. 2022), audio generation (Goel et al. 2022; Ho et al. 2022; Kong et al. 2020), path planning (SHI et al. 2024), trajectory generation (Gu et al. 2022) and spatio-temporal point processes (Yuan et al. 2023). In this paper, we are the fist to introduce the diffusion model to map matching.

Preliminaries

Definitions

Definition 1 (Road Network)**.** *The road network is represented as a directed graph $G = (V, E)$, where V is a set of nodes and E is a set of directed edges. Each node $v_i \in V$ represents an intersection or a end point of a road. Each edge $e = (u, v) \in E$ is a road segment from the entrance node u to the exit node v. $|V|$ denotes the number of intersections and $|E|$ denotes the number of road segments.*

Definition 2 (Trajectory)**.** *A trajectory T is defined as a sequence of GPS points with timestamps, i.e., $T = (p_1, p_2, ..., p_l)$, where l is the trajectory length. Each GPS point $p_i = (lat, lng, t) \in T$ consisting of latitude lat, longitude lng and timestamp t.*

Diffusion and Shortcut

Diffusion (Ho, Jain, and Abbeel 2020) and flow-matching (Lipman et al. 2022) models approach the generative modeling problem by learning an ordinary differential equation (ODE) that transforms noise into data. We consider flow-matching as a special case of diffusion modeling (Kingma and Gao 2023) and use the terms interchangeably. Recently, a new family of denoising generative models called shortcut models overcomes the large number of sampling steps required by diffusion and flow-matching models by introducing desired step size d into the models. We define x_t as a linear interpolation between a data point $x_1 \sim \mathcal{D}$ and a noise point $x_0 \sim \mathcal{N}(0, \mathbb{I})$ of the same dimensionality. The velocity v_t is the direction from the noise to the data point:

$$x_t = (1 - t)x_0 + tx_1 \quad \text{and} \quad v_t = x_1 - x_0. \tag{1}$$

Flow-matching models learn a neural network to estimate the expected value $\overline{v}_t = \mathbb{E}[v_t|x_t]$ that averages over all possible velocities at x_t. Conditioning on step size d, shortcut models refer to the normalized direction from x_t towards the correct next point x'_{t+d} as $s(x_t, t, d)$:

$$x'_{t+d} = x_t + s(x_t, t, d)d. \tag{2}$$

At $d \to 0$, shortcut is equivalent to the flow. Shortcut models have an inherent self-consistency property, namely that one step equals two consecutive steps of half the size:

$$s(x_t, t, 2d) = s(x_t, t, d)/2 + s(x'_{t+d}, t + d, d)/2. \tag{3}$$

This allows shortcut models to be trained using self-consistency targets for $d > 0$ and using the flow-matching loss as a base case for $d = 0$. Therefore the shortcut models can be optimized by the combined shortcut model target:

$$\mathbb{E}_{x_0,x_1}[\|s_\theta(x_t, t, 0) - (x_1 - x_0)\|^2 + \|s_\theta(x_t, t, 2d) - s_t\|^2], \tag{4}$$

where $s_t = s_\theta(x_t, t, d)/2 + s_\theta(x'_{t+d}, t+d, d)/2$ and $x'_{t+d} = x_t + s_\theta(x_t, t, d)d$.

Problem Statement

Given a road network $G = (V, E)$ and a trajectory $T = (p_1, ..., p_l)$, map matching is to map $(p_1, ..., p_l)$ to a sequence of edges in G, denoted as $R = (e_1, ..., e_l)$, where $e_i \in E$. We refer to R as the route, which is essentially a sequence of matched road segments.

Methodology

Figure 2 illustrates the overall framework of DiffMM that contains two key modules: the trajectory encoder and the backbone DiT block (Peebles and Xie 2022). The Trajectory Encoder learns an effective representation $\boldsymbol{C}$ of the input trajectory by point representation and segment representation. To address the limitation of sparse trajectory, point representation adopts Transformer encoder to obtain the sequential dependencies of trajectory. To reduce the harm of noise, segment representation considers the nearby segments as candidates and utilizes attention mechanism to fusion the candidate segment embeddings. Then $\boldsymbol{C}$ serves as one of the conditions for the shortcut model in the denoising process. We choose DiT Block as the backbone of shortcut model to further utilize the sequential dependencies of trajectory, which is helpful for alleviating the problem of sparse trajectory. We first introduce the details of the trajectory encoder. Then we formulate the shortcut for DiffMM and present the architecture of the DiT Block. Finally, we describe how DiffMM is trained and the inference process of it.

Trajectory Encoder

To address the limitations mentioned above, we design an attention-based Trajectory Encoder which encodes raw GPS data, sequential information, directional information, and road network information together to obtain an effective trajectory representation for shortcut model. The input of the encoder is the trajectory $T = (p_1, ..., p_l)$ and the road network $G = (V, E)$ and the output is $\boldsymbol{C} \in \mathbb{R}^{l \times d_{cond}}$, where l is the length of the trajectory and d_{cond} is the dimension of the condition to be used in DiT block.

Point Representation. To address the limitation of low sampling rate, Trajectory encoder obtain the representation of the raw GPS sequence (trajectory points) utilizing the raw GPS data and sequential information. Initially, each point p_i can be represented as a three-element-vector $\boldsymbol{p}_i^{(0)}$ containing its min-max normalized latitude, longitude and timestamp. Then $\boldsymbol{p}_i^{(0)}$ is fed into a fully connected network to get $\boldsymbol{p}_i^{(1)}$ by $\boldsymbol{p}_i^{(1)} = \boldsymbol{p}_i^{(0)}\mathbf{W}_1 + \mathbf{b}_1$, where $\mathbf{W}_1 \in \mathbb{R}^{3 \times d_{emb}}$ and $\mathbf{b}_1 \in \mathbb{R}^{d_{emb}}$ are the learnable parameters.

After obtaining the embeddings of all GPS points $\boldsymbol{P'} = [\boldsymbol{p}_1^{(1)}, ..., \boldsymbol{p}_l^{(1)}]$ in the trajectory, we adopt Transformer encoder (Vaswani 2017) to capture the sequential dependencies in the trajectory:

$$\boldsymbol{P} = TransEncoder(\boldsymbol{P'}). \tag{5}$$

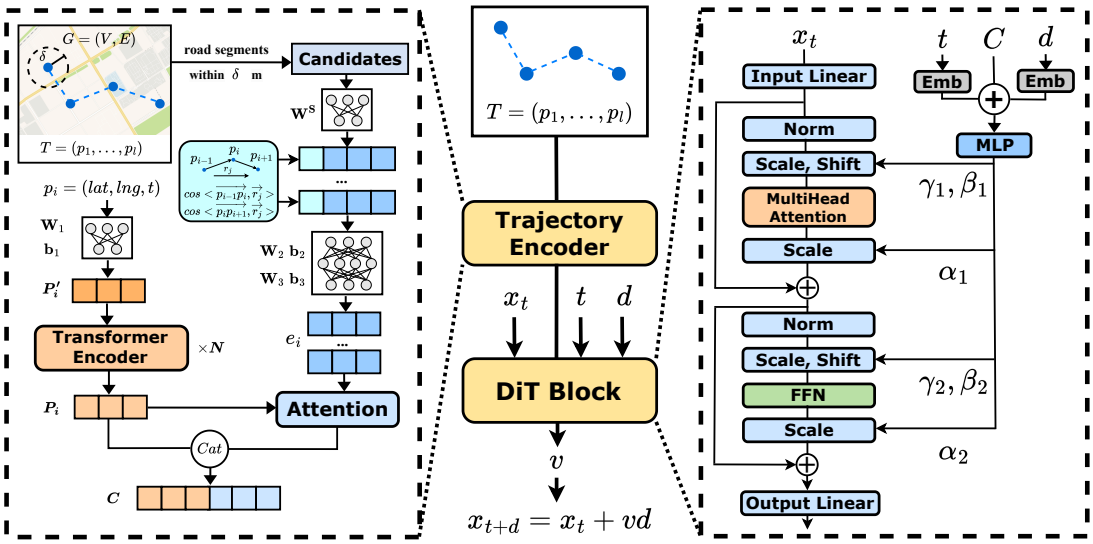


Figure 2: The overview of DiffMM.

A Transformer encoder layer consists of two sub-layers: a multi-head self-attention and a feed-forward network (FFN). Multi-head attention is given by:

$$
\begin{aligned}
&MultiHeadAttn(\mathbf{Q},\mathbf{K},\mathbf{V}) = concat(head_1,...head_h)\mathbf{W}^O,\\
&head_i = Attn(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V),\\
&Attn(\mathbf{Q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V},
\end{aligned}
\tag{6}
$$

where $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are the query, key and value matrix, $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ are parameters for the i-th head, h is the number of attention heads, $\mathbf{W}^O$ is parameter for output.

An FFN is two-layer MLP with ReLU activation:

$$
FFN(\mathbf{X}) = ReLU(\mathbf{X}\mathbf{W}+\mathbf{b})\mathbf{W}'+\mathbf{b}', \tag{7}
$$

where $\mathbf{W}, \mathbf{b}, \mathbf{W}'$ and $\mathbf{b}'$ are learnable parameters.

Let $\mathbf{X}$ be the input, a Transformer layer applies $MultiHeadAttn$ to $\mathbf{X}$ itself followed by an FFN, both with residual connection and layer normalization:

$$
\begin{aligned}
&TransEncoder(\mathbf{X}) = LayerNorm(\mathbf{X}' + FFN(\mathbf{X}')),\\
&\mathbf{X}' = LayerNorm(\mathbf{X} + MultiHeadAttn(\mathbf{X},\mathbf{X},\mathbf{X})).
\end{aligned}
\tag{8}
$$

Segment Representation. To reduce the effects of noisy records, we locate the road segments within δ meters from p_i via R-tree (Guttman 1984) data structure as candidate segments for each GPS point p_i in the trajectory. We then incorporate the candidate segments as road network context to build more informative representation of the trajectory.

For each segment r_{ij} of the candidate segments of p_i, we first apply a fully-connected layer to embed the segment r_{ij} to a high dimension space, i.e., $\mathbb{R}^{d_{emb}}$.

$$
\mathbf{e}_{r_{ij}}^{(0)} = \mathbf{1}_{r_{ij}}\mathbf{W}^{\mathbf{S}}, \tag{9}
$$

where $\mathbf{W}^{\mathbf{S}} \in \mathbb{R}^{|E|\times d_{emb}}$ is learnable parameter, $|E|$ represents the number of road segments, $\mathbf{1}_{r_{ij}} \in \{0,1\}^{|E|}$ is the one-hot vector representing r_{ij} with all elements are 0 except 1 at the position corresponding to the segment id of r_{ij}.

To integrate the direction information, we view each segment r_{ij} as a vector and calculate two cosine similarity: 1) its similarity with the vector from p_{i-1} to p_i, 2) its similarity with the vector from p_i to p_{i+1}. Also, we calculate the distance between p_i and its projection on the road segment r_{ij}, and concatenate these values with $\mathbf{e}_{r_{ij}}^{(0)}$ to get $\mathbf{e}_{r_{ij}}^{(1)} \in \mathbb{R}^{d_{emb}+3}$. We produce the final segment embedding $\mathbf{e}_{r_{ij}}$ for r_{ij} through a Multi-layer Perceptron (MLP):

$$
\mathbf{e}_{r_{ij}} = ReLU(\mathbf{e}_{r_{ij}}^{(1)}\mathbf{W}_2+\mathbf{b}_2)\mathbf{W}_3+\mathbf{b}_3, \tag{10}
$$

where $\mathbf{W}_2 \in \mathbb{R}^{(d_{emb}+4)\times d_{emb}}$, $\mathbf{b}_2 \in \mathbb{R}^{d_{emb}}$, $\mathbf{W}_3 \in \mathbb{R}^{d_{emb}\times d_{emb}}$ and $\mathbf{b}_3 \in \mathbb{R}^{d_{emb}}$ are learnable parameters.

We have obtained the candidate segment embeddings $[R_1,...,R_l]$ for the trajectory, where $R_i \in \mathbb{R}^{n\times d_{emb}}$, n is the number of candidate segments of p_i. However, the number of road segments within δ meters from each p_i could be very different. To address this, we design an attention mechanism to fuse the candidate segments embeddings reasonably to obtain the segment representation. Intuitively, the attention weight for the candidate segment r_{ij} are relative to the GPS point p_i and itself. Therefore, the fusion of the candidate segments embeddings of p_i are as follows:

$$
\begin{aligned}
&\mu_{j,i} = ReLU(concat(\boldsymbol{P}[i], e_j)\mathbf{W}_4+\mathbf{b}_4)\mathbf{W}_5+\mathbf{b}_5,\\
&w_{j,i} = \frac{exp(\mu_{j,i})}{\sum_{\forall s\in C_i} exp(\mu_{s,i})},\\
&f_i = \sum_{\forall j\in C_i} w_{j,i}\cdot e_j,
\end{aligned}
\tag{11}
$$

where $\mathbf{W}_4 \in \mathbb{R}^{2d_{emb}\times d_a}$, $\mathbf{b}_4 \in \mathbb{R}^{d_a}$, $\mathbf{W}_5 \in \mathbb{R}^{d_a\times 1}$ and

$\mathbf{b}_5 \in \mathbb{R}^1$ are learnable parameters, $\boldsymbol{P}[i]$ represents the point representation of p_i, C_i is the candidate segments of p_i and e_j represents the embedding of road segment j.

The final trajectory representation $\boldsymbol{C}$ is the concatenation of point representation and segment representation for each point in the trajectory, i.e., $\boldsymbol{C} = [c_1, ..., c_l]$, $c_i = Concat(\boldsymbol{P}[i], f_i)$, $\boldsymbol{C} \in \mathbb{R}^{l \times d_{cond}}$, $d_{cond} = 2d_{emb}$, and l is the length of the trajectory.

DiffMM Shortcut Model

To further address the problem of low sampling ratio, we use a shortcut model (Frans et al. 2024) to learn the distribution conditioned on the trajectory representation $\boldsymbol{C}$ generated by trajectory encoder. Let x_1 represents the flow target given the trajectory T, $x_1 \in \mathbb{R}^{l \times |E|}$ where $|E|$ is the number of road segments and l is the length of the trajectory. $x_1[i, j]$ is set to 1 if p_i's ground truth matched segment is j otherwise 0. Let $x_0 \in \mathbb{R}^{l \times |E|} \sim \mathcal{N}(0, \mathbb{I})$ represents the start Gaussian noise, value of $x_0[i][j]$ represents the probability that *i-th* point match on segment *j*. We define x_t as a linear interpolation between x_1 and x_0, the shortcut with condition embedding $\boldsymbol{C}$ generated from the original trajectory as follows:

$$x_{t+d} = x_t + s(x_t, t, d, C)d. \tag{12}$$

It also has the self-consistency property:

$$s(x_t, t, 2d, C) = s(x_t, t, d, C)/2 + s(x_{t+d}, t + d, d, C)/2. \tag{13}$$

In this way, we incorporate C into the shortcut model to obtain the conditional distribution. Also, DiffMM can be trained in the shortcut's manner. We choose DiT Block (Peebles and Xie 2022) as the backbone of the shortcut, it first maps input x_t to the model dimension d_{model} through a linear and embeds the time t and the desired step size d by sinusoidal embedding as follows:

$$SinEmb(t) = \begin{cases} cos(t/10000^{\frac{j-1}{d}}) & if\ j\ is\ odd \\ sin(t/10000^{\frac{j-1}{d}}) & if\ j\ is\ even, \end{cases} \tag{14}$$

where d denotes the embedding dimension.

The condition for DiT is $cond = C + SinEmb(t) + SinEmb(d)$, and the multi-head attention part is calculated by the following steps:

$$\begin{aligned} &\alpha_1, \beta_1, \gamma_1 = MLP(cond), \\ &x'_t = \gamma_1 Norm(x_t) + \beta_1, \\ &x = x_t + \alpha_1 MultiHeadAttn(x'_t, x'_t, x'_t). \end{aligned} \tag{15}$$

The FFN part is similar as the multi-head attention part, then the final output is mapped to the original dimension by the output linear for the shortcut process.

Training and Inference

Algorithm 1: DiffMM Training

Input: Trajectories with matched-segments D, road network $G = (V, E)$

while not converged **do**
  $T, x_1 \sim D, x_0 \sim \mathcal{N}(0, \mathbb{I}), (d, t) \sim p(d, t)$
  $x_t \leftarrow (1 - t)x_0 + tx_1$
  $C = TrajEncoder_\phi(T, G)$
  for first k batches **do**
    $s_{target} \leftarrow x_1 - x_0$
    $d \leftarrow 0$
  end for
  for other batches **do**
    $s_t \leftarrow s_\theta(x_t, t, d, C)$
    $x_{t+d} \leftarrow x_t + s_t d$
    $s_{t+d} \leftarrow s_\theta(x_{t+d}, t + d, d, C)$
    $s_{target} \leftarrow (s_t + s_{t+d})/2$
  end for
  Take gradient descent on ϕ and θ by Equation (18)
end while

Algorithm 2: DiffMM Inference

Input: Trajectory T, road network G, sample step M
Output: Matched segments R

$x \sim \mathcal{N}(0, \mathbb{I})$, $d \leftarrow 1/M$, $t \leftarrow 0$
$C = TrajEncoder_\phi(T, G)$
for $i \in [0, ..., M - 1]$ **do**
  $x \leftarrow x + s_\theta(x, t, d, C)d$
  $t \leftarrow t + d$
end for
$R \leftarrow []$
for $i \in [0, ..., T.length]$ **do**
  $R.append(Argmax(x[i]))$
end for
return R

Training. DiffMM is trained by two loss functions: shortcut loss and cross entropy loss. The shortcut loss is defined as follows:

$$\mathcal{L}_{st} = \|s_\theta(x_t, t, 2d, C) - s_{target}\|^2. \tag{16}$$

It is flow-matching target when $d = 0$, otherwise it is the self-consistency target.

We also add an auxiliary cross entropy loss for training:

$$\mathcal{L}_{ce} = CrossEntropy(x_1, x_t + s_t), \tag{17}$$

where x_1 represents the target segments.

The total loss of DiffMM is:

$$\mathcal{L} = \mathcal{L}_{st} + \mathcal{L}_{ce}. \tag{18}$$

We train the overall framework (trajectory encoder and DiT Block) in an end-to-end manner, the pseudo code of the training is shown in Algorithm 1.

Inference. During inference, DiffMM first generates condition embedding by Trajectory Encoder, then obtain the conditional distribution x by shortcut model. We treat the

Dataset	Porto	Beijing
Number of trajectories	1,013,437	1,176,097
Time interval (s)	15	60
Area size (km^2)	11.7×5.2	29.6×30.0
Number of segments	11,491	65,276
Number of intersections	5,330	28,738

Table 1: Dataset statistics.

most possible matched segment of each point i as the matched result, therefore the matched segment of point P_i is $Argmax(x[i])$. We present the pseudo code of the inference procedure in Algorithm 2.

Experiments

In this section, we perform experiments on DiffMM. We first introduce the experimental setup, then evaluate the effectiveness of map matching. Furthermore, we analyze the efficiency of DiffMM during training and inference. In addition, we conduct ablation study on key modules. Finally, we evaluate the robustness of DiffMM by reducing the training data size.

Experimental Setup

Datasets. Table 1 lists the details of the two real-world trajectory data on Porto (PT) and Beijing (BJ), both of which are taxi trajectories. Table 1 also provides the number of segments and intersections and the area size of road networks. Since map matching in urban areas is typically more significant and difficult, we select the central urban area in Porto as the training data, which has relatively smaller area size and fewer number of road segments. To demonstrate the scalability of DiffMM, we select a region in Beijing with large area size and large number of road segments. We obtain the road networks from OpenStreetMap.

For every trajectory, we randomly sample the GPS points in it to generate its sparse trajectory. The average time interval of the sparse trajectory T is t/r where t is the original interval and r is a ratio in $(0, 1)$. For Porto dataset, we set r to 0.2, 0.1, 0.05 and 0.025, whose average interval are 75s, 150s, 300s and 600s respectively. We set r to 0.5, 0.3, 0.2 and 0.1 for Beijing dataset, the average interval are 120s, 200s, 300s and 600s respectively. For each dataset, we randomly split it into training, validating and testing sets with ratio in 40%, 30% and 30%.

Baselines. To evaluate the performance of out model, we include the following methods as baselines: 1) HMM (Newson and Krumm 2009), the most commonly-used method for map matching. 2) DeepMM (Feng et al. 2022), an end-to-end deep learning method based on Recurrent Neural Network (RNN). 3) GraphMM (Liu et al. 2024), a graph-based method which incorporate graph neural networks. 4) RNTrajRec (Chen et al. 2023), a method for trajectory recovery, we set its ratio to 1 to use it for map matching.

Implementations. In our method, we set search dist δ for trajectory encoder to 50 meters. We set the embedding dimensionality d_{emb} to 128 and the condition dimensionality d_{cond} is 256 correspondingly. We stack two layers transformer encoder with four heads for trajectory encoder. For the DiT blocks, we set the hidden dimension d to 512 and stack two layers of DiT blocks. We train the shortcut with $d \in \{1, 1/2\}$ and inference with one step ($M = 1$). The learning rate is set to 1e-3. Our method is implemented in Python 3.11 with PyTorch 2.4.0. All experiments are conducted on a Linux machine with a single NVIDIA RTX 3090 GPU with 24GB memory.

Evaluate Metric. We evaluate all methods' effectiveness through the accuracy of map matching. Let $R = [r_1, r_2, ..., r_l]$ be the ground-truth matched road segments of Trajectory T whose length is l, and $\hat{R} = [\hat{r}_1, \hat{r}_2, ..., \hat{r}_l]$ be the matched road segments predicted by the model. We calculate accuracy as follows:

$$Accuracy(R, \hat{R}) = \frac{1}{l} \sum_{i=1}^{l} \mathbf{1}\{r_i = \hat{r}_i\}. \tag{19}$$

We calculate the accuracy for each trajectory and report the average accuracy over all testing trajectories.

Overall Performance

Table 2 shows the overall performance of models on accuracy. Numbers in bold font indicate the best performers, and underlined numbers represent the next best performers. We observe that DiffMM outperforms all baseline models on the two datasets.

GraphMM has poor performance because the road graph is not constructed correctly.

From the result, we have the following observations:

The sparsity of trajectory destroy the performance significantly. As shown in Table 2, all methods suffer from degraded performance when the trajectory become more sparse, especially HMM, which is very sensitive to the sparsity of trajectory. For example, in Porto dataset, HMM's accuracy is 83.82% when average interval is 150 seconds, however, it decreases to 40.04% when average interval is 600 seconds, which decreases by 43.78 in percentage. Over all methods, DiffMM suffers least from the sparsity.

DiffMM has significant improvement in the two datasets when the records are sparse. Our model achieves the best accuracy on Porto and Beijing datasets, furthermore, it has remarkably significant improvement in accuracy compared to the second-best method when the GPS records are sparse. For example, in Beijing dataset, when the sample ratio is 0.1 (average interval is 600 seconds), DiffMM is 15.28 higher in percentage than the second-best method DeepMM.

Efficiency Study

Table 3 reports the average inference time per 1000 trajectories for map matching and the average training time per epoch in Beijing dataset with sample ratio $r = 0.1$. Bold denotes the best result and underline denotes the second-best

Methods	Porto				Beijing			
	$r = 0.2$	$r = 0.1$	$r = 0.05$	$r = 0.025$	$r = 0.5$	$r = 0.3$	$r = 0.2$	$r = 0.1$
HMM	<u>92.46</u>	<u>83.82</u>	66.62	40.04	<u>89.19</u>	<u>77.51</u>	68.24	46.46
GraphMM	52.84	49.22	37.67	34.49	40.96	20.57	16.32	12.02
DeepMM	86.38	83.68	<u>81.37</u>	<u>78.69</u>	76.59	73.19	<u>71.64</u>	<u>68.25</u>
RNTrajRec	79.56	77.57	75.81	73.76	74.45	69.78	68.68	68.18
DiffMM (Ours)	**93.43**	**91.47**	**89.08**	**86.87**	**90.32**	**88.45**	**87.65**	**85.39**

Table 2: Evaluation of prediction accuracy. A larger value is better (in percentage). Bold denotes the best result and underline denotes the second-best result.

Methods	Inference (s)	Training (min)
HMM	<u>20.57</u>	-
GraphMM	62.79	26.28
DeepMM	88.82	**9.07**
RNTrajRec	627.65	868.23
DiffMM	**1.18**	<u>10.66</u>

Table 3: Inference and training time.

Variants	$r = 0.5$	$r = 0.3$	$r = 0.2$	$r = 0.1$
w/o Trans	90.06	88.33	87.12	84.89
w/o Attn	88.79	87.25	85.70	82.71
w/o Shortcut	89.67	87.92	86.84	83.53
DiffMM	**90.32**	**88.45**	**87.65**	**85.39**

Table 4: Ablation studies on Beijing dataset by accuracy.

result. HMM does not have training time because it does not need training. As shown, our method, DiffMM, is significantly faster than other baseline models by orders of magnitude during inference. For example, DiffMM cost only 1.18 seconds per 1000 trajectories while the second-best method requires 20.57 seconds, which is about a 17-fold speedup. The training time required by DiffMM is about equal to the fastest baseline model, but DiffMM is much faster during inference and significantly outperforms the baseline model in accuracy. The efficiency of DiffMM in both inference and training demonstrates that it is efficient and practical in practice.

Ablation Study

To prove the effectiveness of the key modules of DiffMM, we create three variants of it. **w/o Trans** replaces the Transformer encoder in point representation with a FFN; **w/o Attn** replaces the attention in segment representation with a simple mean calculation; **w/o Shortcut** adopts traditional diffusion instead of the shortcut. We evaluate these variants on Beijing dataset by accuracy and report the results in Table 4.

We observe a drop in performance after removing the Transformer encoder of point representation, and the sparser the trajectory, the more accuracy it dropped. This demonstrates that the Transformer encoder successfully captures the context of sparse trajectory. In addition, we observe a significant drop in performance after removing the attention mechanism of segment representation, which demonstrates that it is helpful to reduce the effects of noise in trajectory. As for shortcut, it additionally considers the desire denoise step size, which is not for the traditional diffusion. This makes shortcut performs better than traditional diffusion in one step denoising. The study results in Table 4 also demonstrate this point, shortcut obtains better results than traditional diffusion.

Training Size	$16,000$	$32,000$	$64,000$	$128,000$
Accuracy	86.01	87.91	89.23	90.03

Table 5: Accuracy Varying Training Data Size.

Robustness of DiffMM

We evaluate the robustness of DiffMM by varying trajectory number of training data. We choose Porto dataset with sample ratio $r = 0.1$ and train our model using 16,000, 32,000, 64,000 and 128,000 trajectories in the training set, respectively. We then test the model on the testing set that have 304,032 trajectories. The accuracy results are reported in Table 5. Typically, reducing the amount of training data degrades the performance of a model. DiffMM's performance does not degrade significantly while training set is shrinking, even if the training size is 16,000, it still outperforms the next best method. This shows that DiffMM extracts the features from trajectory and road network effectively.

Conclusion

We propose *DiffMM*, an encoder-diffusion-based map matching framework. It generates joint embedding of the trajectory and candidate road segments and uses it as conditioning context for one step diffusion process through shortcut model to realize map matching. Extensive experiments on large-scale trajectory datasets highlight the effectiveness and efficiency of our method compared to other methods, particularly for sparse trajectories and complex road network topologies. As for future work, it is feasible to modify the model to generate a dense sequence of road segments through the conditioned diffusion process, which is the task also known as trajectory recovery, whose target is to recover a reasonable dense trajectory for the given sparse trajectory.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (62472174, 62402082), the Open Research Fund of Key Laboratory of Advanced Theory and Application in Statistics and Data Science–MOE, ECNU, the Scientific and Technological Research Program of Chongqing Municipal Education Commission (KJQN202400637), and the Fundamental Research Funds for the Central Universities.

References

Austin, J.; Johnson, D. D.; Ho, J.; Tarlow, D.; and Van Den Berg, R. 2021. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 17981–17993.

Chen, Y.; Zhang, H.; Sun, W.; and Zheng, B. 2023. RNTrajRec: Road Network Enhanced Trajectory Recovery with Spatial-Temporal Transformer. In *International Conference on Data Engineering*, 829–842.

Dhariwal, P.; and Nichol, A. 2021. Diffusion models beat gans on image synthesis. In *Advances in neural information processing systems*, 8780–8794.

Feng, J.; Li, Y.; Zhao, K.; Xu, Z.; Xia, T.; Zhang, J.; and Jin, D. 2022. DeepMM: Deep Learning Based Map Matching With Data Augmentation. *IEEE Transactions on Mobile Computing*, 21(7): 2372–2384.

Frans, K.; Hafner, D.; Levine, S.; and Abbeel, P. 2024. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*.

Goel, K.; Gu, A.; Donahue, C.; and Ré, C. 2022. It's raw! audio generation with state-space models. In *International Conference on Machine Learning*, 7616–7633.

Gong, S.; Li, M.; Feng, J.; Wu, Z.; and Kong, L. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

Gong, Y.-J.; Chen, E.; Zhang, X.; Ni, L. M.; and Zhang, J. 2018. AntMapper: An Ant Colony-Based Map Matching Approach for Trajectory-Based Applications. *IEEE Transactions on Intelligent Transportation Systems*, 19(2): 390–401.

Gu, T.; Chen, G.; Li, J.; Lin, C.; Rao, Y.; Zhou, J.; and Lu, J. 2022. Stochastic trajectory prediction via motion indeterminacy diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 17113–17122.

Guttman, A. 1984. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD international conference on Management of data*, 47–57.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. In *Advances in neural information processing systems*, 6840–6851.

Ho, J.; Salimans, T.; Gritsenko, A.; Chan, W.; Norouzi, M.; and Fleet, D. J. 2022. Video diffusion models. In *Advances in Neural Information Processing Systems*, 8633–8646.

Huang, X.; Li, Y.; Wang, Y.; Chen, X.; Xiao, Y.; and Zhang, L. 2018. CTS: A Cellular-based Trajectory Tracking System with GPS-level Accuracy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4).

Jain, J.; Bagadia, V.; Manchanda, S.; and Ranu, S. 2021. NeuroMLR: Robust & Reliable Route Recommendation on Road Networks. In *Advances in Neural Information Processing Systems*, 22070–22082.

Kingma, D.; and Gao, R. 2023. Understanding diffusion objectives as the elbo with simple data augmentation. In *Advances in Neural Information Processing Systems*, 65484–65516.

Kong, Z.; Ping, W.; Huang, J.; Zhao, K.; and Catanzaro, B. 2020. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.

Li, X.; Thickstun, J.; Gulrajani, I.; Liang, P. S.; and Hashimoto, T. B. 2022. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, 4328–4343.

Lipman, Y.; Chen, R. T.; Ben-Hamu, H.; Nickel, M.; and Le, M. 2022. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

Liu, Y.; Ge, Q.; Luo, W.; Huang, Q.; Zou, L.; Wang, H.; Li, X.; and Liu, C. 2024. GraphMM: Graph-Based Vehicular Map Matching by Leveraging Trajectory and Road Correlations. *IEEE Transactions on Knowledge and Data Engineering*, 36(1): 184–198.

Lou, Y.; Zhang, C.; Zheng, Y.; Xie, X.; Wang, W.; and Huang, Y. 2009. Map-matching for low-sampling-rate GPS trajectories. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 352–361.

Mohamed, R.; Aly, H.; and Youssef, M. 2017. Accurate Real-time Map Matching for Challenging Environments. *IEEE Transactions on Intelligent Transportation Systems*, 18(4): 847–857.

Newson, P.; and Krumm, J. 2009. Hidden Markov map matching through noise and sparseness. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 336–343.

Peebles, W.; and Xie, S. 2022. Scalable Diffusion Models with Transformers. *arXiv preprint arXiv:2212.09748*.

Qiu, X.; Hu, J.; Zhou, L.; Wu, X.; Du, J.; Zhang, B.; Guo, C.; Zhou, A.; Jensen, C. S.; Sheng, Z.; and Yang, B. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. *Proceedings of the VLDB Endowment*, 17(9): 2363–2377.

Qiu, X.; Wu, X.; Lin, Y.; Guo, C.; Hu, J.; and Yang, B. 2025. DUET: Dual Clustering Enhanced Multivariate Time Series Forecasting. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1185–1196.

Rasul, K.; Seward, C.; Schuster, I.; and Vollgraf, R. 2021. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, 8857–8868.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

SHI, D.; Tong, Y.; Zhou, Z.; Xu, K.; Wang, Z.; and Ye, J. 2024. GRAPH-CONSTRAINED DIFFUSION FOR END-TO-END PATH PLANNING. In *International Conference on Representation Learning*, 17723–17741.

Sinha, A.; Song, J.; Meng, C.; and Ermon, S. 2021. D2c: Diffusion-decoding models for few-shot conditional generation. In *Advances in Neural Information Processing Systems*, 12533–12548.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, 2256–2265.

Song, J.; Meng, C.; and Ermon, S. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Stenneth, L.; Wolfson, O.; Yu, P. S.; and Xu, B. 2011. Transportation mode detection using mobile phones and GIS information. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 54–63.

Tashiro, Y.; Song, J.; Song, Y.; and Ermon, S. 2021. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, 24804–24816.

Vaswani, A. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Yang, C.; and Gidofalvi, G. 2018. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *International Journal of Geographical Information Science*, 32(3): 547–570.

Yuan, J.; Zheng, Y.; Xie, X.; and Sun, G. 2013. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE Transactions on Knowledge and Data Engineering*, 25(1): 220–232.

Yuan, Y.; Ding, J.; Shao, C.; Jin, D.; and Li, Y. 2023. Spatio-temporal diffusion point processes. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3173–3184.

Zheng, K.; Zheng, Y.; Xie, X.; and Zhou, X. 2012. Reducing uncertainty of low-sampling-rate trajectories. In *International Conference on Data Engineering*, 1144–1155.