

TRACE: Trajectory Recovery with State Propagation Diffusion for Urban Mobility

Jinming Wang
University of Exeter
Department of Computer Science
Exeter, United Kingdom
jw1294@exeter.ac.uk

Hai Wang
Jingdong Logistics
Beijing, China
hai@seu.edu.cn

Hongkai Wen
University of Warwick
Department of Computer Science
Coventry, United Kingdom
hongkai.wen@warwick.ac.uk

Geyong Min
University of Exeter
Department of Computer Science
Exeter, United Kingdom
G.Min@exeter.ac.uk

Man Luo*
University of Exeter
Department of Computer Science
Exeter, United Kingdom
M.Luo@exeter.ac.uk

Abstract

High-quality GPS trajectories are essential for location-based web services and smart city applications, including navigation, ride-sharing and delivery. However, due to low sampling rates and limited infrastructure coverage during data collection, real-world trajectories are often sparse and feature unevenly distributed location points. Recovering these trajectories into dense and continuous forms is essential but challenging, given their complex and irregular spatio-temporal patterns. In this paper, we introduce a novel diffusion model for **TRA**jectory **rE**covery named **TRACE**, which reconstruct dense and continuous trajectories from sparse and incomplete inputs. At the core of TRACE, we propose a State Propagation Diffusion Model (SPDM), which integrates a novel memory mechanism, so that during the denoising process, TRACE can retain and leverage intermediate results from previous steps to effectively reconstruct those hard-to-recover trajectory segments. Extensive experiments on multiple real-world datasets show that TRACE outperforms the state-of-the-art, offering >26% accuracy improvement without significant inference overhead. Our work strengthens the foundation for mobile and web-connected location services, advancing the quality and fairness of data-driven urban applications. Code is available at: <https://github.com/JinmingWang/TRACE>

CCS Concepts

• **Information systems** → **Incomplete data**; *Location based services*; Temporal data.

Keywords

Diffusion Models, Trajectory Recovery, Location-based Services, Urban Mobility

*Corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License. *WWW '26, Dubai, United Arab Emirates.*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2307-0/2026/04
<https://doi.org/10.1145/3774904.3792461>

ACM Reference Format:

Jinming Wang, Hai Wang, Hongkai Wen, Geyong Min, and Man Luo*. 2026. TRACE: Trajectory Recovery with State Propagation Diffusion for Urban Mobility. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, April 13–17, 2026, Dubai, United Arab Emirates. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3774904.3792461>

1 Introduction

The proliferation of GPS-enabled devices has woven a digital fabric over our urban landscapes, generating vast streams of trajectory data. Location-based web services and smart city applications rely on high-quality trajectories - for navigation, ride-hailing, last-mile delivery, and traffic management. The promise of these technologies - optimized routes, efficient traffic management, and equitable service access - hinges on the availability of continuous, high-quality trajectory data. In the wild, however, trajectories recorded by mobile and edge devices are often *sparse* and *irregular*: sampling rates may drop to save energy, infrastructure coverage is uneven across regions, and device storage constraints limit logging. As a result, real-world trajectories are frequently sampled at low rates, yielding spatially skewed location points with large, uneven time gaps. As shown in Fig. 1, such fragmented paths can misrepresent actual movement, degrading downstream task accuracy and fairness.

To bridge this gap, trajectory recovery serves as an essential pre-processing step, aiming to reconstruct dense and uniform trajectories from sparse and incomplete observations. This task is naturally *conditional generation*: given i) a sparse observation and ii) a set of query timestamps, infer the missing locations, so as to obtain a complete and dense trajectory. While various methods have been developed, they face significant limitations. Map-matching techniques [15, 16] are constrained by the availability of detailed digital road networks, which is not always accessible everywhere, while sequence models like RNNs [6, 24] often suffer from error accumulation over long, unobserved segments. Transformer-based architectures [23, 27] show promise by capturing long-range dependencies, but still struggle when spatio-temporal distribution of input locations is highly skewed. Generative models like VAEs [2, 10, 19] and GANs [5, 9] have framed recovery as a conditional generation task, thus improving fidelity of the generated trajectories, but VAEs can over-smooth or suffer posterior collapse, GANs risk mode

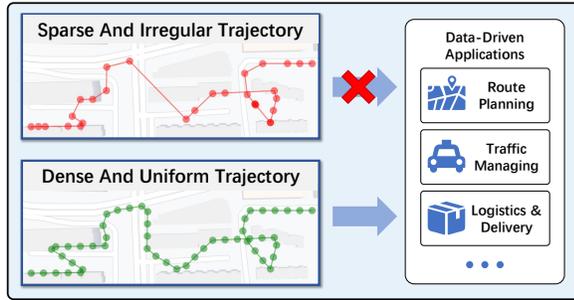


Figure 1: The trajectories collected from mobile edge devices are usually irregular and sparse, which fail to precisely describe the behavior of moving agents, thus cannot be used in data-driven applications.

collapse and unstable - and neither explicitly aggregates evidence across long, irregular gaps.

Recently, diffusion models [7, 17] have emerged as a dominant force in generative modeling, demonstrating remarkable success in related tasks including trajectory generation [26, 28, 29] and recovery of time-series [12].

Applying them naively to trajectory recovery, however, overlooks domain-specific irregularities: unlike typical time-series data, observations (i.e. of agent locations) arrive at uneven, often long time gaps, are spatially constrained by road topology and agent kinematics, and the query timestamps to be filled rarely align with the observed ones. In standard diffusion, each denoising step treats the input almost in isolation: a UNet-style denoising network [18] repeatedly re-encodes the same sparse evidence and static conditions (timestamps, masks, coarse priors) without carrying forward the intermediate features, i.e. what was already inferred. Early steps - dominated by low SNR noise - tend to wash out weak cues from distant observations; later steps then lack a consolidated memory to resolve long, ambiguous gaps, leading to over-smoothed trajectory segments, temporal jitter, or hallucinated detours - especially under fast sampling schedules.

To overcome these challenges, we introduce TRACE, a novel diffusion-based trajectory recovery framework specifically engineered for the irregular spatio-temporal structure of trajectory data. At the core of TRACE is the State Propagation Diffusion Model (SPDM), which fundamentally redesigns the denoising processes to be *memoryful* rather than step-wise isolated. Given sparse observations and a set of query timestamps, TRACE first builds a unified conditioning context that fuses observation/query masks, time gaps, and lightweight priors (e.g., linear interpolation for easy segments; optional context such as day-of-week or user/route IDs). The proposed SPDM then performs conditional denoising while carrying a compact, multi-scale hidden state across diffusion steps, i.e., its stateful denoising network acts as a memory mechanism, allowing multi-scale spatio-temporal features to be retained, accumulated, and reused across successive denoising steps. This propagated state summarizes geometry and motion cues discovered early and feeds them forward, turning a sequence of isolated denoising steps into a coherent, memoryful refinement process. Concretely, in TRACE

a UNet backbone processes the conditioning context, while a step-aware state propagation module updates the hidden state to concentrate capacity on long, uncertain gaps and avoid re-processing those stable regions. This not only avoids redundant computation but also provides richer, evolving context, i.e., knowledge from previous denoising steps can now serve as auxiliary conditions for subsequent steps, empowering the model to reconstruct challenging, irregular segments with far greater accuracy. The technical contributions of this paper are summarized as follows:

- To the best of our knowledge, we are the first to formalize trajectory recovery as conditional generation under severe spatial/temporal irregularity, and propose a novel diffusion-based framework, TRACE, that is explicitly designed to handle the sparse and irregular nature of real-world trajectory data. This provides a more reliable data foundation for a broad spectrum of downstream applications, including location-based web services.
- We introduce the State Propagation Diffusion Model (SPDM), a novel architecture that augments the standard diffusion models with a step-aware state propagation mechanism. SPDM carries compact, multi-scale states across the denoising process - by reusing this intermediate knowledge as conditioning, SPDM avoids redundant processing but concentrates capacity on the hard-to-recover segments while skipping the already stable ones.
- We evaluate TRACE across multiple real-world datasets and sparsity regimes. Extensive experiments show that TRACE consistently outperforms strong baselines by significant margins. Ablations confirm that the proposed SPDM mechanism is the key, achieving up to a 26.65% improvement in recovery accuracy over the standard diffusion model, establishing a new state-of-the-art without significant inference overhead.

2 Preliminaries

2.1 Problem Formulation

Timestamps: Let $S = \{s_1, s_2, \dots, s_N\}$ denote a set of N timestamps where $s_i \in \mathbb{R}^+$ represents the exact time of a GPS recording. Timestamps are ordered ($s_1 < s_2 < \dots < s_N$) but are usually irregularly spaced, reflecting real-world constraints such as energy-saving sampling or signal loss.

Trajectory: Given timestamps S , a corresponding trajectory $\tau^S = \{p_{s_1}, p_{s_2}, \dots, p_{s_N}\}$ is a sequence of GPS coordinates, where each point p_{s_i} records the longitude and latitude of the moving agent at time s_i be a location point recording longitude and latitude.

Sample Interval: For consecutive timestamps s_i and s_{i+1} , the sample interval $\Delta s_i = s_{i+1} - s_i$ quantifies temporal sparsity. Small and constant sample intervals computed from S and τ^S indicate a dense and uniform trajectory, otherwise the trajectory is sparse and irregular.

Query: The query $Q = \{q_1, q_2, \dots, q_M\}$ specifies M timestamps where the corresponding GPS coordinates $\{p_{q_1}, p_{q_2}, \dots, p_{q_M}\}$ are to be inferred. Note that these coordinates form a complementary trajectory denoted as τ^Q . Non-trivial trajectory recovery occurs when $Q \cap S = \emptyset$, requiring inference of locations at unobserved times.

Trajectory Recovery: Given a trajectory τ with large or irregular sample intervals, its corresponding S , a query Q , and other types

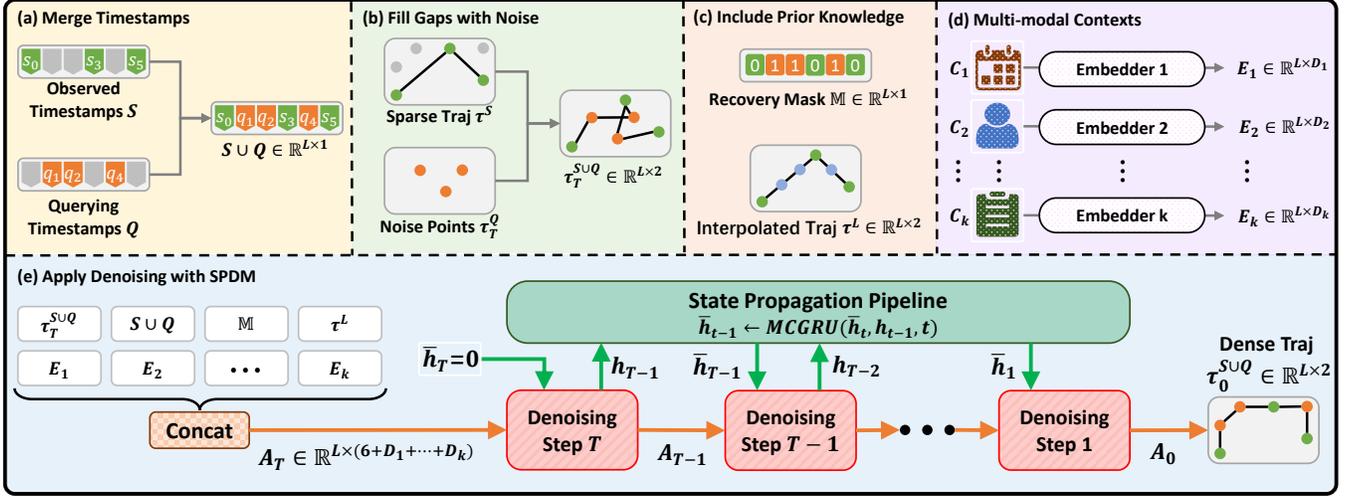


Figure 2: TRACE Framework. (a) Merge querying timestamps into observed timestamps in chronological order. (b) Initialize missing points with Gaussian noise. (c) Prior knowledge are included to explicitly tell the model where to recover. (d) The embedding of multi-modal contextual information. (e) The denoising process with SPDM that generates the dense trajectory.

of auxiliary contexts C such as date, user ID and waybills. The objective of the trajectory recovery task is to find a function that produces prediction $\hat{\tau}^Q$ and minimizes the error between $\hat{\tau}^Q$ and the ground-truth τ^Q .

2.2 Diffusion Models

Denoising diffusion probabilistic models (DDPM) [7] is constructed on the foundation of a forward Markov chain, which progressively corrupts clean data by adding Gaussian noise at each discrete step. The model trains a denoising neural network to accurately predict the noise introduced at each step of the forward process. Once trained, the network can be used to generate clean data such as trajectory points, starting from pure Gaussian noise.

Given a clean data sample x_0 drawn from the training distribution $x_0 \sim p_{\text{data}}(\mathbf{x})$. The forward Markov process is defined by a noise schedule $\beta_1, \beta_2, \dots, \beta_T$, specifying the variance of Gaussian noise added at each time step. At step t , the noisy data x_t is obtained through a conditional distribution:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t \mathbf{I}) \quad (1)$$

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{\beta_t}\varepsilon_{t-1:t}$$

where $\alpha_t = 1 - \beta_t$ and $\varepsilon_{t,t+1} \sim \mathcal{N}(0, 1)$ is single-step noise added. Alternatively, x_t can be expressed directly in terms of the clean data initial x_0 as:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}) \quad (2)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon_{0:t}$$

where $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$, the notation $\varepsilon_{0:t} \sim \mathcal{N}(0, 1)$ is the multi-step noise, and x_1, x_2, \dots, x_T represent the intermediate states. As T becomes sufficiently large, the forward process effectively transforms the data distribution $p_{\text{data}}(x_0)$ into a standard Gaussian distribution.

To enable the reverse process, the model learns a denoising network parameterized by θ , which approximates the reverse transition distribution $p_\theta(x_{t-1}|x_t)$ by minimizing the divergence from the true posterior $q(x_{t-1}|x_t, x_0)$. This is achieved by training a noise predictor $\varepsilon_{0,t,\theta}(x_t, t)$, which estimates the noise $\varepsilon_{0:t}$ introduced at each step.

During inference, the generation process involves iterative sampling of the learned reverse transitions p_θ , guided by the denoising network ε_θ . Additionally, the framework can incorporate conditional signals to improve generation quality. For instance, when conditioning on auxiliary information c , the model can be extended to train a conditional denoising network $\varepsilon_{0,t,\theta}(x_t, t, c)$ using an analogous loss function, thus enhancing the fidelity and control of the generated data.

There exist other formulations of diffusion processes that attempt to improve sampling efficiency through fewer generation steps. Such models include DDIM [20], score-based models [21], or PNDM [11]. However, as long as the model still involves a multi-step denoising procedure, the redundancy persists, and they can be used jointly with our SPDM.

3 Methodology

3.1 TRACE Overview

Trajectory recovery requires the integration of heterogeneous data, including trajectories, timestamps, contextual metadata (e.g., date and trip duration), and other user-specific features, which must be processed to a unified representation compatible with diffusion frameworks. These input data — serving as the condition for trajectory recovery — are illustrated in Fig. 2(a-d), and the details will be discussed in Sec. 3.2. The aggregated representation is then fed into the denoising process, shown in Fig. 2(e), which iteratively reconstructs the querying locations by removing noise from an initial Gaussian noise. Crucially, TRACE replaces the conventional

denoising pipeline with SPDM to address the spatio-temporal irregularity problem. It introduces a state propagation mechanism that propagates multi-scale spatio-temporal features across denoising steps. The architecture of SPDM, including the specially designed neural networks used in each denoising step and the state propagation pipeline, will be explained in Sect. 3.3. Lastly, the training procedure for TRACE is very different from the training of traditional diffusion models. This difference stems from the unique inter-step dependency introduced by the state propagation pipeline, and Sec. 3.4 will present the details of new training algorithm.

3.2 Condition Aggregation Stage

To enable robust trajectory recovery under spatio-temporal irregularities, we integrate various prior knowledge to guide the recovery process. The sparse trajectory τ^S and associated timestamps S are the primary observation that the recovery process can rely on, while the query timestamps Q specifies target timestamps with unknown coordinates τ^Q , which is initialized as Gaussian noise. The query timestamps can be manually or procedurally selected, ensuring highly flexible and customizable recovery pattern. To synchronize observation and query, we merge S and Q into a chronologically ordered sequence denoted $S \cup Q$ of length L . Similarly, we construct an assembled trajectory $\tau^{S \cup Q}$ following the same chronological order. To explicitly indicate which coordinates in $\tau^{S \cup Q}$ should undergo recovery, we label the missing point indices with a binary mask $\mathbb{M} \in \{0, 1\}^L$, where $\mathbb{M}_k = 1$ indicates that the k -th timestamp in $S \cup Q$ comes from Q and the k -th coordinate in $\tau^{S \cup Q}$ comes from τ^Q . Moreover, we realize that simple linear interpolation is sufficient to recover some trajectory segments, e.g., straight paths, smooth and dense trajectory segments. Therefore, we introduce a linearly interpolated trajectory $lerp(\tau)$ over $S \cup Q$ to augment the input. Concretely, it fills the unknown coordinates with linearly interpolated values inferred from nearby points, providing the model with a geometrically plausible prior.

Industrial applications usually provide a variety of heterogeneous contexts $\{C_1, C_2, \dots, C_K\}$ associated with trajectories, e.g. weekday, date, user ID, waybill details. Since the embedding part is not the focus of our method, we assume that one module is specially designed for each context, where each module consists of several simple operations such as linear or convolutional layers. In the end, each context C_i is converted to a d_i dimensional sequential feature $E_i \in \mathbb{R}^{L \times d_i}$ aligned with the trajectories.

At the end of this stage, the final aggregated feature tensor $A \in \mathbb{R}^{L \times D}$ is constructed as:

$$A \leftarrow \text{Concatenate}(\tau^{S \cup Q}, S \cup Q, \mathbb{M}, lerp(\tau), E_1, \dots, E_K) \quad (3)$$

where D is the final dimensionality of the features. To conclude, this formulation ensures sparse and irregular query regions, enriched with various prior knowledge and contextual information, enabling adaptive recovery of complex trajectory segments.

3.3 Denoising With SPDM

Traditional diffusion models iteratively denoise the input noise x^T over T steps, transforming it into clean data x^0 . In the proposed framework, our input is denoted as A_T , which will be recovered to A_0 through intermediate steps $\{A_{T-1}, A_{T-2}, \dots, A_1\}$. In fact, although

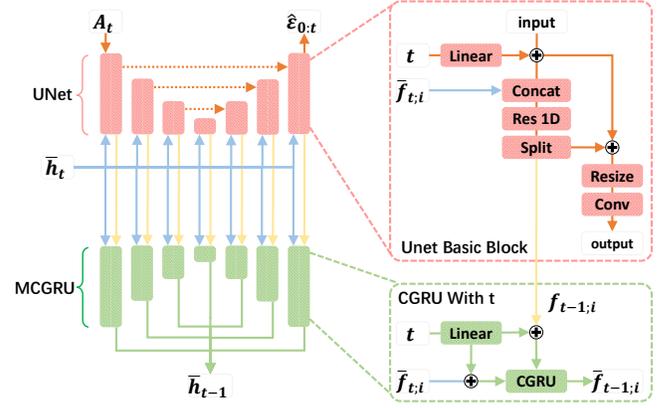


Figure 3: The architecture of the modified UNet (upper, red) and the proposed MCGRU (lower, green).

A_t is the input for step t , the denoising operation only applies to τ_t^Q , which is a portion inside A_t corresponding to the noisy coordinates, while the other portion of A_t is fixed throughout the entire denoising process.

A novel formulation for the denoising step is essential to enable effective state propagation in SPDM. We begin by defining the states: let h_t be the **single-step** hidden state that encapsulates the local features extracted at step t , and let \tilde{h}_t be the **multi-step** hidden state containing all useful knowledge from previous steps $[t + 1, T]$. To enhance feature richness and reusability, we define both single- and multi-step hidden states as multi-scale sequential features, i.e. $h_t = \{f_{t,1}, f_{t,2}, \dots, f_{t,b}\}$ and $\tilde{h}_t = \{\tilde{f}_{t,1}, \tilde{f}_{t,2}, \dots, \tilde{f}_{t,b}\}$, where b is the number of sequential features in the state. Under this formulation, for each denoising step t , the denoising network takes A_t , \tilde{h}_t and t as inputs and produces h_{t-1} along with the predicted noise $\hat{\epsilon}_{0,t}$. Subsequently, the state propagation network accepts \tilde{h}_t , h_{t-1} and t to output the updated state \tilde{h}_{t-1} .

Based on this formulation, our neural network architecture is implemented accordingly. The denoising network in TRACE is built upon a UNet architecture, which inherently supports multi-scale feature extraction and aligns well with our state definitions. To accommodate state propagation, the UNet is modified as illustrated in the upper part of Fig. 3. Specifically, the UNet comprises exactly b basic blocks — matching the number of features in the states. Building on top of traditional UNet blocks, each modified block i also processes additional input $\tilde{f}_{t,i} \in \tilde{h}_t$ to generate the corresponding output feature $f_{t-1,i}$. Finally, all output features of the b UNet blocks are assembled to h_{t-1} .

After we get h_{t-1} generated by the denoising UNet, the state propagation network is responsible for updating the cumulative knowledge and producing \tilde{h}_{t-1} . Concretely, we implement a novel Multi-scale Convolutional Gated Recurrent Unit (MCGRU) with diffusion step awareness, shown in the lower part of Fig. 3. MCGRU contains b CGRU blocks in parallel, mirroring the UNet’s hierarchy. Building on top of traditional CGRU, the modified version also takes into account the diffusion time t , which enhances its compatibility with the multi-step denoising pipeline.

3.4 Training of TRACE

In traditional diffusion model training for trajectory recovery, a dense trajectory with corresponding timestamps is provided, which are then divided into the observation part (S, τ^S) and the query part (Q, τ_0^Q). With a randomly selected time step $t \in [1, T]$, the forward diffusion process is applied to generate τ_t^Q by adding noise $\varepsilon_{0:t} \in \mathcal{N}(0, 1)$. Then, the denoising network is trained to predict $\varepsilon_{0:t}$. However, during TRACE training, the SPDM denoising step t requires the state \bar{h}_t from previous iterations. This disables random step sampling and necessitates a sequential training paradigm, where each data sample iterates through all steps from $t = T$ to $t = 1$, so that \bar{h}_t is always available.

The above solution makes the training of the denoising network possible, yet we also have to train the state propagation network, i.e., each state \bar{h}_t must contain useful knowledge for the subsequent steps. This requirement imposes two sub-constraints: i) the denoising network at step t should output appropriate h_{t-1} and ii) the state propagation network should effectively aggregate or filter knowledge to yield beneficial \bar{h}_{t-1} . Intuitively, the ideal strategy is to jointly train the entire denoising process, with the overall objective being:

$$\min_{\theta} \sum_{t=1}^T \text{MSE}(\bar{\varepsilon}_{0:t}, \varepsilon_{0:t}) \quad (4)$$

where θ represents the learnable parameters of the models. However, incorporating all tens to hundreds of denoising steps in each training iteration is computationally impractical. Instead, we partition the denoising process into smaller segments comprising several consecutive steps, e.g. including two steps in one training iteration:

$$\min_{\theta} (\text{MSE}(\hat{\varepsilon}_{0:t}, \varepsilon_{0:t}) + \text{MSE}(\hat{\varepsilon}_{0:t-1}, \varepsilon_{0:t-1})) \quad (5)$$

Another challenge in training TRACE arises from the noise sampling process. The standard diffusion model training algorithm samples $\varepsilon_{0:t} \sim \mathcal{N}(0, 1)$ independently in each training iteration, because it does not involve the complex inter-step dependencies introduced by the state-propagation mechanism. In contrast, our algorithm iterates t from T to 1 for each data sample, and one training iteration involves at least two denoising steps. Consequently, the training of TRACE cannot ignore the dependency among noises, i.e., multi-step noises $\varepsilon_{0:t}$ and $\varepsilon_{0:t-1}$. To derive a set of dependent noise samples, we first sample a list of single-step noises $\{\varepsilon_{0:1}, \varepsilon_{1:2}, \dots, \varepsilon_{T-1:T}\}$, where each $\varepsilon_{t-1:t} \sim \mathcal{N}(0, 1)$ is always independent. Then, starting with A_0 , we recursively apply single-step diffusion forward using Eq. 1 to obtain A_1 to A_T . Next, we apply following equation to compute the multi-step noise $\varepsilon_{0:t}$ for each t :

$$\varepsilon_{0:t} = \frac{A_t - \sqrt{\bar{\alpha}_t} A_0}{\sqrt{1 - \bar{\alpha}_t}} \quad (6)$$

This equation is a changed form of Eq. 2, it ensures that the dependency among the noises is respected within each training iteration, as well as during the entire life cycle of a data sample. Now, the general principles of training TRACE have finally been established, and a naive procedure of the proposed training algorithm is presented in Algorithm 1.

Algorithm 1 Training One Sample (2-step)

```

0: Input:  $S, Q, \tau^S, \tau_0^Q, \text{lerp}(\tau), \mathbb{M}, C_1, C_2, \dots, C_K$ .
1: Generate  $\{\varepsilon_{t-1:t} | t \in [1, T]\}$ .
2: Derive  $\{\varepsilon_{0:t} | t \in [1, T]\}$  with Eq. 6.
3: Apply diffusion to  $\tau_0^Q$  to get  $\{\tau_t^Q | t \in [1, T]\}$ .
4:  $h_T \leftarrow 0$ 
5: for  $t \leftarrow T$  to 1 do
6:    $E_i \leftarrow \text{Embed}_i(C_i)$  for  $i \in \{1, \dots, K\}$ .
7:   Obtain  $A_{t-1}, A_t$  using concatenation as in Eq. 3.
8:   // Train the first step
9:    $\hat{\varepsilon}_{0:t}, h_{t-1} \leftarrow \text{UNet}(A_t, \bar{h}_t, t)$ 
10:  // Update hidden state
11:   $\bar{h}_{t-1} \leftarrow \text{MCGRU}(\bar{h}_t, h_{t-1}, t)$ 
12:  // Train the second step
13:   $\hat{\varepsilon}_{0:t-1}, h_{t-2} \leftarrow \text{UNet}(A_{t-1}, \bar{h}_{t-1}, t-1)$ 
14:  Compute loss with Eq. 5. and do optimization.
15: end for

```

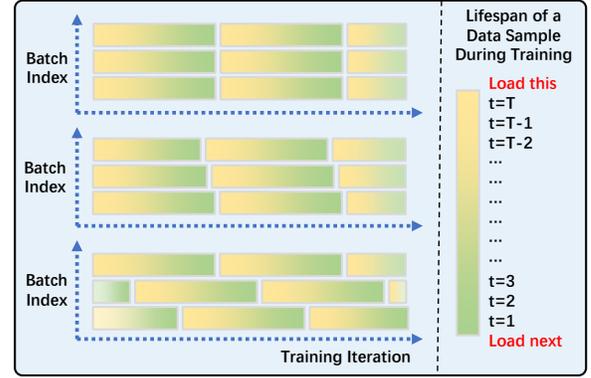


Figure 4: Three ways of batch management. Top: Shared t among all samples in a batch. Middle: Different t for each sample with little offset. Bottom: Uniformly distributed t over the range $[1, T]$.

Although the above design is sufficient for training, practical implementation demands additional techniques. As mentioned above, each data sample in the training batch undergoes T training iterations with a synchronized countdown of t across the entire batch (Fig. 4, top). Consequently, the model overfits to a narrow range of denoising steps and cannot generalize to the full range of t . To address this, we implement a dynamic batch management technique that assigns a private t to each sample in a batch, with an offset among them (Fig. 4, middle). We can further promote generalization across the entire range of t by uniformly assigning $t \in [1, T]$ to each data sample, as illustrated in Fig. 4 bottom. In this scheme, the t of each sample is updated independently, and a new sample is loaded in-place once its t reaches 0.

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate TRACE on three datasets. Two of them comprise dense and smooth taxi trajectories collected in Xi'an city and Chengdu city, China. To assess the model's performance across

| Metric | Method | Dataset | | |
|--------------------------------|----------------|--------------|--------------|---------------|
| | | Xi'an | Chengdu | Logistics |
| MSE ↓ ($\times 10^{-3}$) | DeepMove | 5.297 | 26.857 | 30.060 |
| | AttnMove | 0.068 | 0.234 | 3.201 |
| | PriSTI | 0.019 | 0.292 | 2.206 |
| | DT + RP | 0.326 | 6.919 | 4.250 |
| | TRACE w/o SPDM | <u>0.017</u> | <u>0.202</u> | <u>2.025</u> |
| | TRACE | 0.010 | 0.159 | 0.449 |
| NDTW ↓ ($\times 10^{-3}$) | DeepMove | 36.295 | 169.470 | 72.814 |
| | AttnMove | 2.257 | 5.458 | 22.652 |
| | PriSTI | 1.174 | 3.975 | 12.656 |
| | DT + RP | 8.473 | 28.773 | 13.006 |
| | TRACE w/o SPDM | <u>1.156</u> | <u>3.809</u> | <u>11.062</u> |
| | TRACE | 1.154 | 3.597 | 5.520 |
| MAE ↓ ($\times 10^{-3}$) | DeepMove | 31.695 | 149.169 | 94.151 |
| | AttnMove | 3.119 | 7.250 | 33.744 |
| | PriSTI | 1.587 | <u>2.397</u> | <u>2.233</u> |
| | DT + RP | 3.195 | 112.212 | 21.458 |
| | TRACE w/o SPDM | <u>1.352</u> | 4.646 | 21.796 |
| | TRACE | 1.335 | 1.725 | 2.121 |

Table 1: The overall comparisons of six models with three metrics on three dataset. The trajectory length is 512 points, with 50% GPS points erased and the models are asked to recover the erased points.

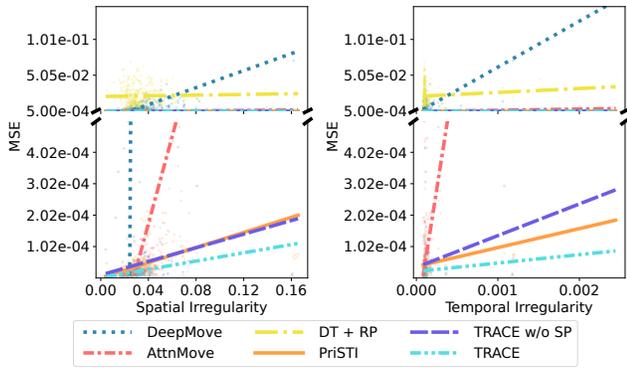


Figure 5: The trend of recovery errors as the trajectory gets more irregular in terms of spatial and temporal distributions. There are breaks in y-axis because the magnitude of MSE differs a lot for different methods. Tested on Xi'an with 70% points erased and trajectory length of 512 points.

diverse mobility patterns and metadata, we also include a dataset from last-mile logistics, capturing courier movements in residential areas with irregular sampling intervals. For conditional contexts and metadata, we extract the trajectory starting date and time, the moving agent’s ID, and trip duration. In addition, the logistics data set includes information for package delivery and receiving. All GPS coordinates undergo z-score normalization, and the time stamp sequence of each trajectory is rescaled to the range [0, 1].

Metrics. Trajectory recovery focuses on the exact spatial similarity between the recovered coordinates and the ground truth ones. Therefore, we adopt three evaluation metrics: The Mean Squared Error (MSE), the Mean Absolute Error (MAE), and the Normalized

Dynamic Time Warping (NDTW). Lower values are preferable for all metrics.

Baselines. We evaluate TRACE against four baselines. i) **DeepMove** [4]: An RNN-based trajectory forecasting model adapted for recovery via auto-regressive imputation. ii) **AttnMove** [27]: A Transformer-based approach fusing intra-/inter-trajectory contexts via self- and cross-attention. iii) **PriSTI** [12]: A diffusion model for time-series imputation, modified for trajectories by treating GPS coordinates as 2D signals. iv) **DT+RP**: Combines DiffTraj [28] (UNet-based trajectory generation) with RePaint [13] (image inpainting via diffusion), which is capable of adapting content generation diffusion models to recovery tasks. Additionally, we ablate TRACE by replacing SPDM with a standard diffusion pipeline (**TRACE w/o SPDM**), which aims to test the effectiveness of SPDM.

4.2 Results

4.2.1 Overall Comparison. Table 1 summarizes the performance of TRACE against the baselines. Non-diffusion methods (DeepMove, AttnMove) exhibit significantly higher errors, underscoring the limitations of auto-regressive and attention-based architectures in handling sparse, irregular trajectories. In contrast, Diffusion-based baselines show marked improvements except for the approach combining DiffTraj with RePaint, which emphasizes the need to design and train diffusion models specific for recovery tasks. The ablation study (TRACE without SPDM) highlights the need for state propagation, while competitive, it underperforms TRACE in most test cases, highlighting the importance of SPDM.

4.2.2 Addressing Spatio-Temporal Irregularity. We prove the effectiveness of SPDM on addressing the challenge of spatio-temporal irregularity inherent to trajectory data. Given a trajectory to be recovered τ^S , we compute two metrics to measure the spatial and temporal irregularities: i) the standard deviation of distances between consecutive point pairs and ii) the standard deviation of sample intervals. A large variance indicates that the trajectory is irregular and skewed; otherwise, the trajectory points are uniformly distributed. The results are shown in Fig. 5. As the trajectory gets increasing irregular, all methods tend to have worse performance. However, the proposed TRACE is apparently more robust to such irregularities. When comparing TRACE with and without SPDM, we can see that the SPDM achieves significant improvement when dealing with irregular trajectories, showcasing the effective and successful design of SPDM.

4.2.3 Scalability. The length and sparsity of trajectories can vary greatly in industrial scenarios. In this experiment, we study the scalability of the proposed method with different trajectory lengths and sparsity levels. As illustrated in Fig. 6, we first select the three strongest methods: PriSTI, TRACE without SPDM, and TRACE. The left column presents three error metrics with trajectory length varying from 64 to 512, and the right column shows the slowly increasing errors as the trajectory gets more sparse. The sparsity is measured by the percentage of points removed from a dense trajectory. As we can see, the plots exhibit parallel trends, while TRACE consistently outperforms other methods.

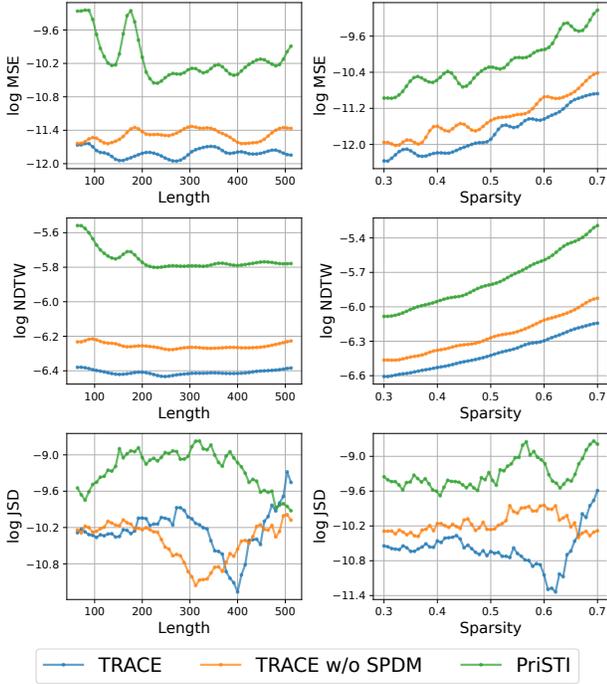


Figure 6: The comparison among sparsity levels and trajectory lengths.

| Method | Steps | Time (s) | MSE $\times 10^{-3}$ | NDTW $\times 10^{-3}$ | MAE $\times 10^{-3}$ |
|----------------------|-------|----------|----------------------|-----------------------|----------------------|
| DDIM | 500 | 119.176 | 0.0432 | 2.01 | 3.068 |
| | 51 | 12.36 | 0.0349 | 2.868 | 2.221 |
| | 26 | 6.65 | 0.0351 | 2.897 | 2.239 |
| | 11 | 3.18 | 0.0490 | 3.083 | 2.365 |
| DDIM With State Prop | 500 | 129.57 | 0.0356 | 2.750 | 2.502 |
| | 51 | 13.02 | 0.0272 | 2.332 | 1.847 |
| | 26 | 7.03 | 0.0256 | 2.320 | 1.822 |
| | 11 | 3.58 | 0.0260 | 2.420 | 1.914 |

Table 2: The comparison of recovery time and quality with and without state propagation pipeline. Tested on Xi’an with 70% points erased and trajectory length of 512 points.

4.2.4 Efficiency. DDIM [20] is a variant in the diffusion model family that can speed up recovery by skipping some intermediate denoising steps, trading result quality for faster recovery. We compare the recovery time and quality of TRACE using DDIM with or without state propagation to demonstrate the superior efficiency of TRACE, and the results are shown in Table 2. The denoising times are recorded with 100 samples in a batch on a NVIDIA GeForce RTX 3070 Ti Laptop GPU. For plain DDIM, the 11 denoising steps version is $119.176/3.18 = 37.48$ faster than the full 500-step version, but the recovery quality is decreased by $(0.0490 - 0.0432)/0.0490 \times 100\% = -11.84\%$. In contrast, building a state propagation pipeline on top of the 11-step DDIM introduces only 12% more parameters and 0.4 seconds of running

| Method | Parameters | MACs | FLOPs |
|-----------------------|--------------|--------------|--------------|
| TRACE | 6.28M | 1.13G | 2.28G |
| TRACE w/o SPDM | 6.17M | 0.99G | 2.01G |
| DT + RP | 9.10M | 0.63G | 1.27G |
| PriSTI | 7.32M | 3.24G | 6.51G |
| AttnMove | 8.15M | 3.94G | 7.92G |
| DeepMove | 10.11M | 1.94G | 9.93G |

Table 3: The number of parameters, multiply-accumulate operations (MACs), and floating point operations (FLOPs) of all methods.

| | Average Speed (m/s) | Moving Distance (km) |
|----------------|---------------------|----------------------|
| Ground Truth | 1 | 1 |
| Sparse Traj | 0.6180 | 0.5788 |
| Recovered Traj | 0.8252 | 0.8480 |

Table 4: The moving speed and distance estimations computed from trajectories after normalization.

time, and also improves the upper bound of recovery quality by $(0.0349 - 0.0256)/0.0349 \times 100\% = 26.65\%$. This indicates that the proposed SPDM can be jointly used with other diffusion sampling techniques that accelerate recovery, while achieving even higher recovery quality.

We also provide a table describing the computational cost of all baselines, shown as Table 3. In particular, we present the number of parameters, multiply-accumulate operations (MACs), and floating point operations (FLOPs) of all baselines. We notice that TRACE is very parameter-efficient due to its efficient design. The SPDM only introduces a small number of parameters and computations to our diffusion model, this further proves its effectiveness. One thing to note is that the diffusion models require iteratively denoising, so the cost should be multiplied by the denoising steps, making them computationally costly compared to non-diffusion-based baselines.

4.2.5 Effectiveness of Training Algorithms. We analyze the effectiveness of the proposed training algorithms and techniques introduced in Sect. 3.4. First, we want to study the effect of different number of denoising steps included in a single training iteration, as shown in Fig. 7a. The result indicates that the number of steps trained in each iteration do not have a great influence on the final performance, and 2-step is preferable since it requires the least training time. Second, we show the importance of dynamic batch management during training. As we can see in Fig. 7b, the uniformly distributed t for each data sample within a batch leads to the best convergence, while a shared t for the entire batch results in the worst recovery quality. This study highlights the necessity of the proposed training algorithm.

4.3 Case Study

A case study was conducted using courier trajectories in a logistics scenario. Analyzing the speed and distance of moving people or vehicles is crucial for many location-based data-driven Web applications, such as navigation and last-mile delivery. The trajectory

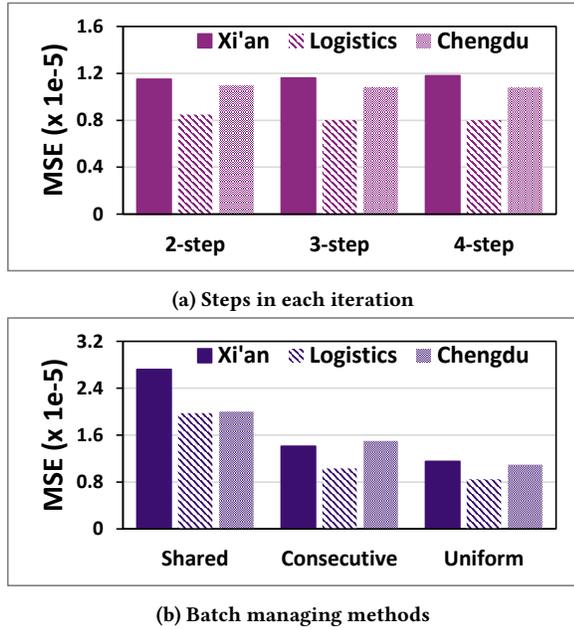


Figure 7: The comparisons among: (a) Number of denoising steps included in single training iteration. (b) Three ways of batch managing introduced in Fig 4. Tested on Xi'an with 70% points erased and trajectory length of 512 points.

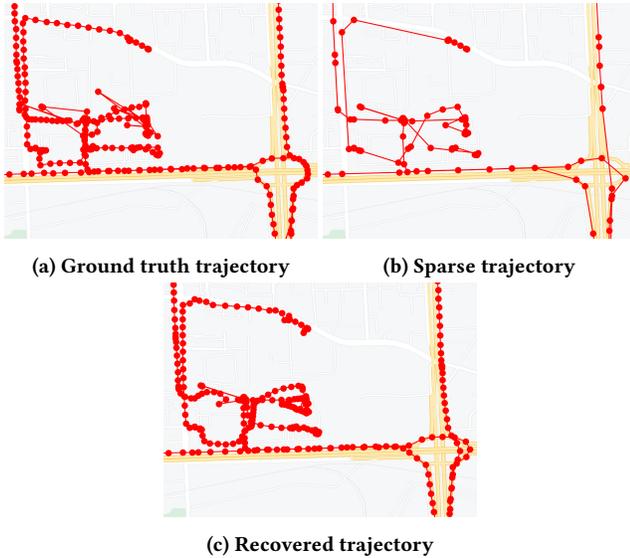


Figure 8: The dataset of: (a) Dense trajectories as ground truth. (b) Trajectories with the same sparsity as usual logistics scenario. (c) Recovered trajectories using TRACE.

of the user is usually collected to compute the speed and the moving distance. However, these trajectories are usually sparse due to constrained device-server communication rates, limited sampling rates, and unstable signal conditions. In addition, storing massive

amounts of trajectories for large-scale web-service can be costly. All these limitations hinder accurate computation of user speed and moving distance.

We collected very dense trajectories in a crowded apartment region. A subset was sampled from each dense trajectory according to a normal last-mile delivery sampling interval, resulting in only 30% of the points remaining. TRACE was trained to recover these sparse trajectories to dense ones. We used the moving distances and speed computed from the very dense trajectories as ground truth and then computed the estimation using the sparse and the recovered trajectories. For more direct comparison, we normalized the ground truth speed to 1 meter per second and the moving distance to 1 kilometer, and the results are shown in Table 4. We observed that the estimations using sparse trajectories captured 60% of the actual speed and distance, while the recovered trajectories increased this percentage to around 83%. An illustration of the trajectory before and after recovery is shown in Figure 8.

5 Related Work

Trajectory Recovery. Numerous efforts have been made to increase the accuracy of trajectory recovery. Certain map-based methods [3, 8, 14, 16] utilize a predefined set of points of interest (POI) or a pre-collected road network as strong prior knowledge. In contrast, DHTR [25] performs free-space trajectory recovery using RNN, which eliminates the need for road network information. However, for very long and sparse trajectories, RNNs struggle to capture dependencies among points. TrajBERT [19] and AttnMove [27] are methods based on Transformers [23], which excel in capturing long-term dependencies within sequence.

Diffusion-based Trajectory Generation/Recovery. Existing approaches have successfully applied diffusion models to generate or recover time series data, including CSDI [22] and SSSD [1]. In particular, PriSTI [12] is a diffusion-based method designed for time series recovery, using spatio-temporal conditions and geographic factors for more accurate recovery. Moreover, another thread of existing work uses diffusion models for trajectory generation rather than recovery. For example, DiffTraj [28] employs UNet [18] to perform denoising steps with several simple contexts such as trajectory length, starting and ending points. More recently, some diffusion-based methods utilize road networks to guide the generation process, such works include Diff-RNTraj [26] and ControlTraj [29].

6 Conclusion

In conclusion, this paper presents TRACE, a diffusion-based framework for robust trajectory recovery. At its core, TRACE incorporates the State Propagation Diffusion Model (SPDM), which introduces a novel state propagation pipeline that enables knowledge sharing across denoising steps. This design effectively addresses the spatio-temporal irregularities of real-world trajectory data, leading to more accurate reconstruction of sparse segments while reducing redundant computation. Extensive experiments on multiple real-world datasets confirm the superior robustness and efficiency of TRACE over state-of-the-art methods. By providing precise and efficient trajectory recovery, our work enhances the data reliability for large-scale location-based web services and smart city applications.

References

- [1] Juan Miguel Lopez Alcaraz and Nils Strodthoff. 2023. Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models. arXiv:2208.09399 [cs.LG]
- [2] Xinyu Chen, Jiajie Xu, Rui Zhou, Wei Chen, Junhua Fang, and Chengfei Liu. 2021. TrajVAE: A Variational AutoEncoder model for trajectory generation. *Neurocomputing* 428 (2021), 332–339. doi:10.1016/j.neucom.2020.03.120
- [3] Yuqi Chen, Hanyuan Zhang, Weiwei Sun, and Baihua Zheng. 2022. RNTrajRec: Road Network Enhanced Trajectory Recovery with Spatial-Temporal Transformer. arXiv:2211.13234 [cs.LG]
- [4] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1459–1468. doi:10.1145/3178876.3186058
- [5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML]
- [6] Alex Graves. 2012. *Supervised Sequence Labelling*. Vol. 385. Springer, Berlin, Heidelberg, 5–13. doi:10.1007/978-3-642-24797-2_2
- [7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. *CoRR* abs/2006.11239 (2020). arXiv:2006.11239 <https://arxiv.org/abs/2006.11239>
- [8] Fujin Hou, Xiaowei Lan, Jingrong Chen, Yuhuan Dong, Xucai Zhuang, and Jianqing Wu. 2021. A Sparse Taxi Trajectory Data Recovery and Calibrate Algorithm. In *2021 China Automation Congress (CAC)*. IEEE, 5414–5419. doi:10.1109/CAC53003.2021.9728679
- [9] Wenjun Jiang, Wayne Xin Zhao, Jingyuan Wang, and Jiawei Jiang. 2023. Continuous Trajectory Generation Based on Two-Stage GAN. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 4 (Jun. 2023), 4374–4382. doi:10.1609/aaai.v37i4.25557
- [10] Diederik P Kingma and Max Welling. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML]
- [11] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. 2022. Pseudo Numerical Methods for Diffusion Models on Manifolds. arXiv:2202.09778 [cs.CV] <https://arxiv.org/abs/2202.09778>
- [12] Mingzhe Liu, Han Huang, Hao Feng, Leilei Sun, Bowen Du, and Yanjie Fu. 2023. PriSTI: A Conditional Diffusion Framework for Spatiotemporal Imputation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1–10. doi:10.1109/icde55515.2023.00150
- [13] Andreas Lugmayr, Martin Danelljan, Andrés Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. 2022. RePaint: Inpainting using Denoising Diffusion Probabilistic Models. *CoRR* abs/2201.09865 (2022), -. arXiv:2201.09865 <https://arxiv.org/abs/2201.09865>
- [14] Junwei Ma, Chao Yang, Shiwen Mao, Jian Zhang, Senthilkumar CG Periaswamy, and Justin Patton. 2022. Human Trajectory Completion with Transformers. In *ICC 2022 - IEEE International Conference on Communications*. IEEE, 3346–3351. doi:10.1109/ICC45855.2022.9838743
- [15] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Seattle, Washington) (GIS '09). Association for Computing Machinery, New York, NY, USA, 336–343. doi:10.1145/1653771.1653818
- [16] Huimin Ren, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li, and Yu Zheng. 2021. MTrajRec: Map-Constrained Trajectory Recovery via Seq2Seq Multi-task Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) (KDD '21). Association for Computing Machinery, New York, NY, USA, 1410–1419. doi:10.1145/3447548.3467238
- [17] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. High-Resolution Image Synthesis with Latent Diffusion Models. *CoRR* abs/2112.10752 (2021), 1–10. arXiv:2112.10752 <https://arxiv.org/abs/2112.10752>
- [18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR* abs/1505.04597 (2015), 1–10. arXiv:1505.04597 <http://arxiv.org/abs/1505.04597>
- [19] Junjun Si, Jin Yang, Yang Xiang, Hanqiu Wang, Li Li, Rongqing Zhang, Bo Tu, and Xiangqun Chen. 2024. TrajBERT: BERT-Based Trajectory Recovery With Spatial-Temporal Refinement for Implicit Sparse Trajectories. *IEEE Transactions on Mobile Computing* 23, 5 (2024), 4849–4860. doi:10.1109/TMC.2023.3297115
- [20] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising Diffusion Implicit Models. *CoRR* abs/2010.02502 (2020), 1–10. arXiv:2010.02502 <https://arxiv.org/abs/2010.02502>
- [21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456* 1 (2020), -.
- [22] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. arXiv:2107.03502 [cs.LG]
- [23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017), 1–10. arXiv:1706.03762 <http://arxiv.org/abs/1706.03762>
- [24] Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, and Kai Feng. 2021. Deep Trajectory Recovery with Fine-Grained Calibration using Kalman Filter. *IEEE Transactions on Knowledge and Data Engineering* 33, 3 (2021), 921–934. doi:10.1109/TKDE.2019.2940950
- [25] Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, and Kai Feng. 2021. Deep Trajectory Recovery with Fine-Grained Calibration using Kalman Filter. *IEEE Transactions on Knowledge and Data Engineering* 33, 3 (2021), 921–934. doi:10.1109/TKDE.2019.2940950
- [26] Tonglong Wei, Youfang Lin, Shengnan Guo, Yan Lin, Yiheng Huang, Chenyang Xiang, Yuqing Bai, Menglu Ya, and Huaiyu Wan. 2024. Diff-RNTraj: A Structure-aware Diffusion Model for Road Network-constrained Trajectory Generation. arXiv:2402.07369 [cs.LG] <https://arxiv.org/abs/2402.07369>
- [27] Tong Xia, Yong Li, Yunhan Qi, Jie Feng, Fengli Xu, Funing Sun, Diansheng Guo, and Depeng Jin. 2023. History-enhanced and Uncertainty-aware Trajectory Recovery via Attentive Neural Network. *ACM Trans. Knowl. Discov. Data* 18, 3, Article 53 (dec 2023), 22 pages. doi:10.1145/3615660
- [28] Yuanshao Zhu, Yongchao Ye, Shiyao Zhang, Xiangyu Zhao, and James J. Q. Yu. 2023. DiffTraj: Generating GPS Trajectory with Diffusion Probabilistic Model. arXiv:2304.11582 [cs.LG]
- [29] Yuanshao Zhu, James Jianqiao Yu, Xiangyu Zhao, Qidong Liu, Yongchao Ye, Wei Chen, Zijian Zhang, Xuetao Wei, and Yuxuan Liang. 2024. ControlTraj: Controllable Trajectory Generation with Topology-Constrained Diffusion Model. arXiv:2404.15380 [cs.LG] <https://arxiv.org/abs/2404.15380>