TrajWeaver: Trajectory Recovery with State Propagation Diffusion Model

Jinming Wang
University of Exeter
jw1294@exeter.ac.uk

Hai Wang
Southeast University & JD Logistic
hai@seu.edu.cn

Hongkai Wen
University of Warwick
hongkai.wen@warwick.ac.uk

Geyong Min
University of Exeter
G.Min@exeter.ac.uk

Man Luo
University of Exeter
M.Luo@exeter.ac.uk

Abstract—With the proliferation of location-aware devices, large amount of trajectories have been generated when agents such as people, vehicles and goods flow around the urban environment. These raw trajectories, typically collected from various sources such as GPS in cars, personal mobile devices, and public transport, are often *sparse* and *fragmented* due to limited sampling rates, infrastructure coverage and data loss. In this context, trajectory recovery aims to reconstruct such sparse raw trajectories into their dense and continuous counterparts, so that fine-grained movement of agents across space and time can be captured faithfully. Existing trajectory recovery approaches typically rely on the prior knowledge of travel mode or motion patterns, and often fail in densely populated urban areas where accurate maps are absent. In this paper, we present a new recovery framework called TrajWeaver based on probabilistic diffusion models, which is able to recover dense and refined trajectories from the sparse raw ones, conditioned on various auxiliary features such as Areas of Interest along the way, user identity and waybill information. The core of TrajWeaver is a novel State Propagation Diffusion Model (SPDM), which introduces a new state propagation mechanism on top of the standard diffusion models, so that knowledge computed in earlier diffusion steps can be reused later, improving the recovery performance while reducing the number of steps needed. Extensive experiments show that the proposed TrajWeaver can recover from raw trajectories of various lengths, sparsity levels and heterogeneous travel modes, and outperform the state-of-the-art baselines significantly in recovery accuracy. Our code is available at: https://anonymous.4open.science/r/TrajWeaver/

Index Terms—Trajectory recovery, diffusion model, urban computing.

Fig. 1. The collected trajectory is too sparse to be used in data-driven applications.

I. INTRODUCTION

The ubiquity of GPS-enabled devices has revolutionized urban data collection, producing vast amounts of trajectory data that serve as the foundation for data-driven smart city applications. For instance, trajectories can be used to discover road networks [1], classify vehicle types [2] and identify transportation mode [3]. However, the collected trajectories are often sparse and disjointed due to constraints such as limited sampling rates and intermittent signal loss. This sparsity poses a significant challenge in accurately capturing the continuous movement of agents across urban landscapes. Trajectory recovery emerges as an essential task, aiming to reconstruct these incomplete trajectories into detailed and seamless paths
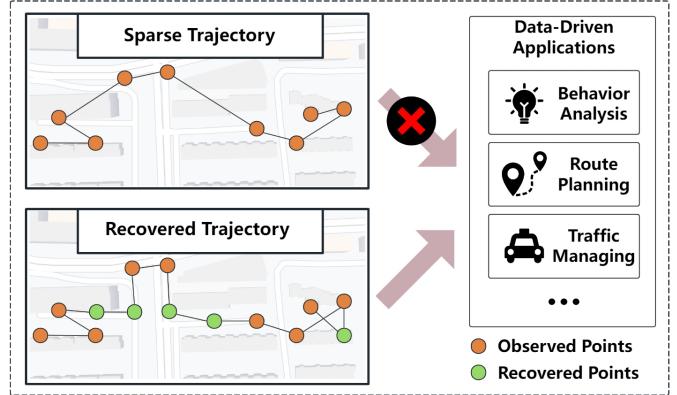
that accurately reflect the agents' movements. Specifically, given a sparse trajectory and various additional contexts such as user ID or waybill data, trajectory recovery is to generate and fill points into the gaps within the sparse trajectory. However, this task is fraught with challenges. First, the moving agents can exhibit heterogeneous mobility patterns. Especially in last-mile delivery scenarios, the couriers can frequently enter or exit apartments, moving quickly or staying at one location for a long time. The second challenge involves the diverse lengths and sparsity levels of trajectories, which requires the recovery model to have good scalability. Lastly, the detailed map information is usually absent in complex metropolitan or residential areas. Most existing methods perform poorly in these scenarios, struggling to generalize across diverse transportation modes or densely populated areas.

Recently, diffusion models have demonstrated robust performance in content recovery tasks such as image inpainting [4] and time series imputation [5]. A diffusion model operates through two multi-step processes. The diffusion process corrupts the data through recursive noise injection, eventually resulting in pure noise. Conversely, the denoising process employs neural networks to progressively remove the noise at each step, thereby recovering the original data. Compared

to other content generation methods like Variational Autoencoders (VAEs) [6], [7] and Generative Adversarial Networks (GANs) [8], [9], diffusion models exhibit superior modeling, pattern capturing and content generation capabilities across a range of tasks [10], [11], [12]. These strengths suggest that diffusion models hold significant potential for enhancing trajectory recovery, offering more accurate and robust reconstructions of trajectories.

Although the unique multi-step procedure endows diffusion models with strong recovery ability, several challenges arise when applying them to trajectory recovery. The first challenge lies in integrating diverse recovery conditions into diffusion models in an economic and efficient way. Different from trajectory generation task which generates the entire trajectory based on few conditions, trajectory recovery generates additional points given various conditions. In addition to the sparse trajectory being the foundation of this recovery task, extra contexts include the prior knowledge such as linear interpolated points, and numerous additional contexts such as user ID, weekday, and waybill information. Given that diffusion models perform recovery through many denoising steps, where conditions are fused at each step, the integration of conditions must be multi-modal, computationally efficient, and sensitive to the spatio-temporal correspondence among these conditions. A common way of conditioning in diffusion models is through cross-attention, as stated in [13], which employs a pre-trained module to integrate and convert conditions into an embedding, then fuse the embedding into the denoising neural network. However, due to the squared memory usage of cross-attention modules, this way of condition fusing is not efficient, especially for long trajectories.

As the basic version of diffusion model, Denoising Diffusion Probabilistic Model (DDPM) [14] often takes thousands of denoising steps to complete recovery. Techniques such as Denoising Diffusion Implicit Model (DDIM) [15], DPM Solver [16] and Pseudo Numerical Methods for Diffusion Models (PNDM) [17] can reduce the number of steps needed to tens, while other methods have accelerated the denoising process through model pruning and distillation [18], [19], [20]. However, most approaches sacrifice recovery quality to achieve lower latency, and it is challenging to simultaneously ensure fast and high-quality recovery. The core issue that prevents efficient recovery is the inherently isolated nature of the denoising steps. The denoising process often takes hundreds of steps, where each step performs extensive feature extraction. However, as defined by diffusion framework, the denoised content is the only information passed to the next step, which hinders the sharing of knowledge among steps. A more efficient recovery process could be realized by promoting greater collaboration among denoising steps, enabling the reuse of features, and thereby achieving high-quality recovery with significantly fewer steps.

Scalability is a critical challenge in applying diffusion models to trajectory recovery, given the varying lengths and sparsity levels of trajectories. In logistics scenarios, trajectory lengths can range from tens of points to thousands, and the total distance can vary from hundred meters to kilometers. Short trajectories often lack sufficient conditional information, which requires the model to accurately reconstruct them with limited information. In the case of sparse trajectories, more points should be generated and inserted into the trajectory, so diffusion models introduce more noise into the data and leads to greater instability. On the other hand, long trajectories require the model to distill useful information at each denoising step, which can increase model complexity and slow down recovery. Moreover, the diverse spatio-temporal patterns correspond to different trajectory length and sparsity can further complicate the condition fusion process. Therefore, ensuring the scalability of diffusion models is essential for their successful application in trajectory recovery.

To address the challenges outlined above, we propose a novel diffusion-based trajectory recovery framework called TrajWeaver, which is capable of effectively aggregate and fuse diverse conditions into the multi-step recovery process. To achieve high-quality and low-latency trajectory recovery simultaneously, TrajWeaver introduces a new variant of the diffusion model, State Propagation Diffusion Model (SPDM). In particular, it implements a state propagation pipeline on top of the standard denoising process, which allows knowledge sharing and features reuse among steps. This inter-step information sharing not only improves recovery efficiency, but also enhances scalability by transmitting multi-scale features within state. The design of TrajWeaver is anchored in three key elements: an advanced module for effective condition aggregation, a neural network that enables state fusing and propagation, and a specialized training algorithm tailored for SPDM. Through extensive experiments, TrajWeaver demonstrates its ability to recover both pedestrian and vehicle trajectories in complex urban environments with high accuracy and scalability.

Overall, our contributions can be summarized as follows:

- We propose a novel diffusion-based trajectory recovery framework named TrajWeaver, which is capable of aggregating various forms of conditions to achieve fast and accurate trajectory recovery in complex environments with dynamic moving patterns.
- We introduce SPDM, which consists a new state propagation mechanism on top of the standard diffusion models, so that more efficient recovery process and better scalability can be achieved.
- We conduct extensive experiments to validate the effectiveness of the proposed method, including comparisons with existing methods, ablation studies and use case analysis.

II. RELATED WORK

Trajectory Recovery. Numerous efforts have been dedicated to increase the accuracy of trajectory recovery. Certain map-based methods [21], [22], [23], [24] utilize a predefined set of points of interest (POI) or a pre-collected road network as strong prior knowledge, but the detailed map information is not always guaranteed. In contrast, DHTR [25] performs

free space trajectory recovery leveraging RNNs, which does not require road network information. However, for very long and sparse trajectories, RNNs struggle to capture dependencies among points. This is because RNNs recovery consecutive points in auto-regressive manner, which may accumulate the recovery errors. AttnMove [26] is a method based on Transformers [27], which excel in capturing long-term dependencies within sequence. In particular, it assumes that the agent moves in periodic manner, thus similar trajectories will be produced in each period. By aggregating all history trajectories and the current trajectory via inter- and intra-trajectory attention, the underlying mobility patterns of the agent can be revealed. TrajBERT [28] is another Transformer-based method that is mainly built on BERT [29], which incorporates multi-aspect spatial-temporal aware designs.

Diffusion-based Trajectory Generation. There are no existing works that apply diffusion models to trajectory recovery, but diffusion-based methods on time series recovery and trajectory generation can be used as references. Existing approaches have successfully applied diffusion models to generate and recover time series data, including CSDI [30] and SSSD [31]. In particular, PriSTI [5] is a recent diffusion-based method designed for time series recovery, it utilizes spatio-temporal conditions and geographical factors for more accurate recovery. On the other hand, another thread of existing work uses diffusion models for trajectory generation rather than recovery. The main difference between these two tasks is that trajectory generation produces the full trajectory without knowing existing portions, and the aim of this task is usually to ensure the generated dataset having similar distribution as the original trajectory dataset. For example, DiffTraj [32] employs UNet [33] to perform denoising steps with several simple contexts such as trajectory length, beginning point and end point. Diff-RNTraj [34] combines the advantages of both PriSTI and DiffTraj to perform trajectory generation with road network integrated.

More Efficient Diffusion Models. Denoising Diffusion Probabilistic Models (DDPM) [14] is the basic version of diffusion model, which requires hundreds of denoising steps to complete recovery. Many techniques were proposed to reduce the redundancy and improve efficiency of DDPM. Such works include Denoising Diffusion Implicit Model (DDIM) [15] and DPM Solver [16], which can shrink the denoising process to 10 to 20 steps with a cost of around 10% to 30% loss in recovery quality. There exists methods capable of distilling denoising process to one step [19], but it sacrifices more result quality for extremely fast recovery. Pseudo Numerical Methods for Diffusion Models (PNDM) [17] is a more advanced diffusion sampler that performs fast denoising process while preserves the original result quality. Moreover, model compression and quantization methods such as Diff-Pruning [18] and PTQD [35] can effectively reduce computation waste thus accelerate each denoising step. DeepCache [20] is the most similar method to our approach, which optimizes denoising process by excluding certain parts of the UNet [33] and reusing previously extracted features.

III. PRELIMINARIES

A. Problem Formulation

Trajectory: A trajectory $\tau = \{p_0, p_1, ...\}$ is a chronologically ordered sequence that describes the movement of an entity, where each element is a spatio-temporal point $p_i = (lng_i, lat_i, time_i)$ consisting of longitude, latitude and time stamp.

Sparse Trajectory: Given a dense trajectory τ, a sparse trajectory $\tilde{\tau}$ contains a subset of τ and fails to accurately describe the continuous movement of the entity in the real world.

Query: A query Q is a manually selected or procedurally generated sequence of time stamps. We want to determine the location of the moving entity at each specified time stamp in Q.

Trajectory Recovery: Given sparse trajectory $\tilde{\tau}$, query Q, and other types of contexts such as date, weekday, user ID, waybills, etc. The goal of trajectory recovery is to accurately predict the unknown locations correspond to the provided Q.

B. Diffusion Models

Diffusion Process. The diffusion process consists of a series of noise addition steps, where each step is formulated as in Equation 1. In this equation, $t \in [0, T)$ denotes the index for the diffusion step, α_t and β_t are noise schedules and $\alpha_t = 1 - \beta_t$. $\varepsilon_{t:t+1}$ represents single step noise drawn from $\mathcal{N}(0, 1)$.

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

We can further derive and compute x_t directly from the original content x_0 instead of applying the forward step t times, as shown in Equation 2. Here, $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$, and $\varepsilon_{0:t+1} \sim \mathcal{N}(0, 1)$ represents the multi-step noise added to x_0 to produce x_t in one step. The notation

$$x_{t+1} = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon_{0:t+1} \tag{2}$$

Denoising Process. The denoising process aims to reconstruct clear content x_0 from random noise x_T through a series of denoising steps. Given noisy content x_{t+1} at step $t+1$ where $t \in [0, T-1]$, the aim of this denoising step is to predict the mean and standard deviation of the less noisy content x_t. The equation below explains one denoising step.

$$
\begin{aligned}
\mu_t &= \frac{x_{t+1} - \beta_t * \varepsilon_{0:t+1}^{pred}}{\sqrt{\bar{\alpha}_t}} \\
\sigma_t &= \sqrt{\frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} * \beta_t} * z, \quad z \sim \mathcal{N}(0, 1) \\
x_t &= \mu_t + \sigma_t
\end{aligned}
\tag{3}
$$

A neural network is trained to predict $\varepsilon_{0:t+1}$ in equation 3 since it is the only unknown value. During training, we choose t from valid range from 0 to $T-1$, then sample $\varepsilon_{0:t} \sim \mathcal{N}(0, 1)$ and apply diffusion forward process to x_0 to obtain x_t, where x_0 is the original content and x_t is the noisy content. The model is trained to produce $\epsilon_{0:t}^{pred}$ and the loss is computed using mean squared error.
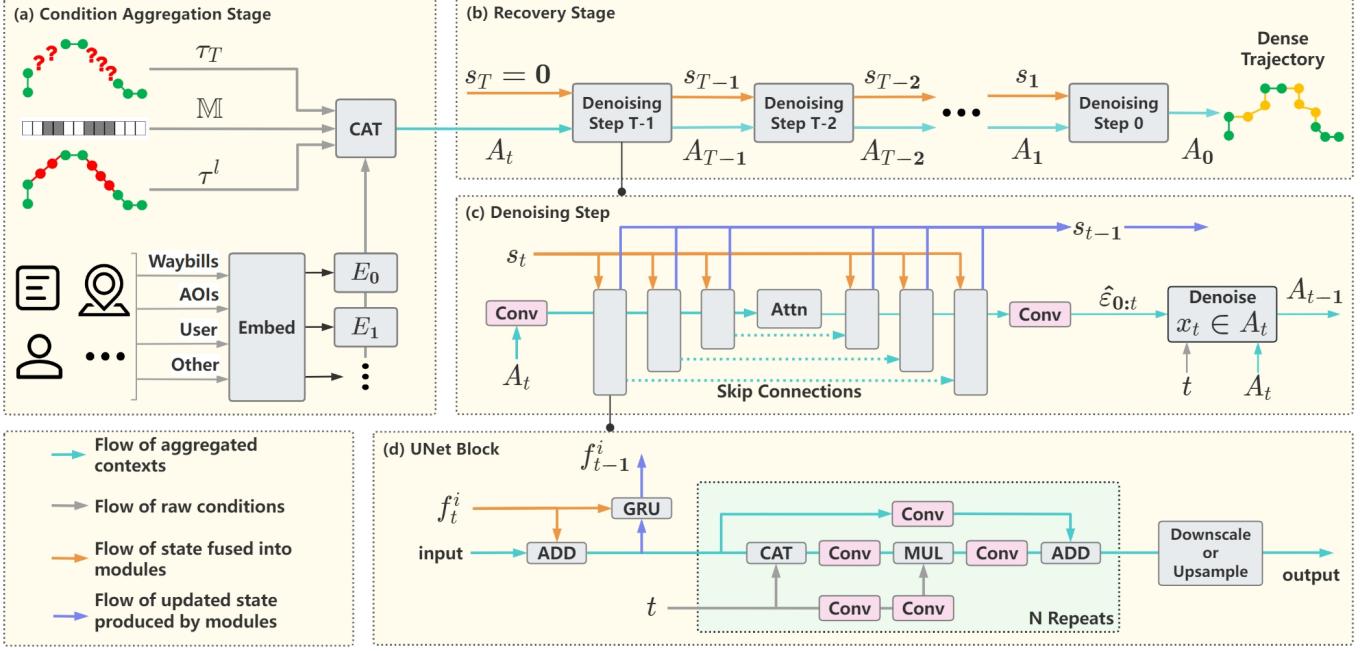
Fig. 2. TrajWeaver Framework. (a): The aggregation of trajectory, prior knowledge and all other contexts. (b): The denoising pipeline for trajectory recovery. (c): The denoising step, including the proposed neural network architecture. (d): A single block in the proposed network.

IV. METHODOLOGY

A. TrajWeaver Overview

Condition Aggregation Stage. Before entering the pipeline of TrajWeaver, we first need to define all input components that are crucial for the model's performance. The main input is a sparse trajectory $\tilde{\tau}$, which represents a sequence of spatio-temporal points sampled at irregular intervals. Alongside this sparse trajectory, a query Q is selected either manually or procedurally, containing a series of time stamps at which the locations need to be predicted. These time stamps from Q are inserted into the sparse trajectory $\tilde{\tau}$, with the corresponding unknown locations initialized with random noise values, denoted as x_T. The outcome of this initialization is the trajectory τ_T, where TrajWeaver's objective is to convert the noisy locations $x_T \in \tau_T$ into accurate predictions $x_0 \in \tau_0$, representing the original, noise-free trajectory. The process of composing τ_T is depicted in Figure 3.

To facilitate the recovery process, we generate a binary mask $\mathbb{M}$, where each entry is set to 0 or 1 to indicate whether a point is part of the original trajectory or an inserted point, respectively. This mask helps the model distinguish between observed and unobserved data points. Furthermore, to incorporate prior spatio-temporal knowledge into the model, we create an additional trajectory representation, τ^l, where unknown locations are filled with linearly interpolated values instead of random noise. This serves as a preliminary estimate or "prior guess" of the true trajectory, providing a useful baseline for the model to build upon during the diffusion process.
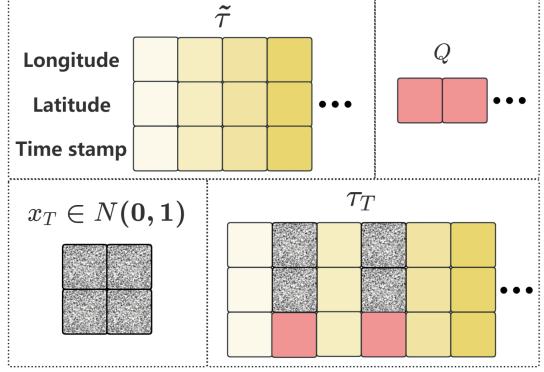

Fig. 3. The trajectory with noise added τ_T is a composition of the given sparse trajectory $\tilde{\tau}$, the query Q and the noise x_T.

In addition to the previous inputs, various contextual features $C_0, C_1, ...$ that are associated with the trajectory are also considered. Due to the sequential form of trajectories, the positions of tokens contain important spatio-temporal knowledge. To ensure good positional correspondence among conditions, we also want to process the contexts $C_0, C_1, ...$ to sequential forms just like other conditions. However, the forms of these contexts can vary widely in different industrial scenarios, a universal embedding module design is inappropriate. Therefore, we assume a specifically designed module $Embed_i$ for each context C_i, which processes it into a sequential embedding E_i of the same length as τ_T. These modules can be as trivial as several convolutional layers or fully-connected layers. All conditions are now in sequential format with the

same length. Instead of applying cross attention to fuse them as in Stable Diffusion [13], we simply concatenate them together, witch keeps positional alignment among sequences, thus ensures good spatio-temporal correspondence. The final aggregated condition is denoted as A_T. This stage is illustrated in Figure 2 (a) and formulated as follows:

$$
\begin{aligned}
A_T &\leftarrow Concat(\tau_T, \tau^l, \mathbb{M}, E_0, E_1, ...) \\
E_i &\leftarrow Embed_i(C_i)
\end{aligned}
\tag{4}
$$

Recovery Stage. TrajWeaver recovery stage focuses on removing noise from $x_T \in A_T$, while all other portions of A_T remain fixed throughout the recovery procedure. As shown in Figure 2 (b), recovery takes T steps counting down from $T-1$ to 0. Each denoising step is illustrated in Figure 2 (c). Given A_t in step t, a neural network is utilized to make a prediction $\hat{\varepsilon}_{0:t}$. Then, x_t is denoised to x_{t-1}, which also transforms A_t to A_{t-1}. A typical denoising step in diffusion models takes only A_t and t as inputs then produces $\hat{\varepsilon}_{0:t}$. As a result, the traditional denoising steps are nearly isolated, and some similar features can be repeated extracted in each step. This significant redundancy makes the denoising process inefficient, and prevents the model from achieving low latency and high recovery quality simultaneously. Therefore, TrajWeaver employs SPDM which allows inter-step information sharing and feature reuse. Specifically, as indicated by the orange arrows in Figure 2 (b), each denoising step produces an additional state s_{t-1} that contains useful information for subsequent steps, and each step also absorbs state s_t from the previous step, with the initial state s_T filled with 0. This state propagation mechanism is completed by the specially designed denoising neural network, as formulated in Equation 5.

$$
\hat{\varepsilon}_{0:t}, s_{t-1} = \mathbf{Net}(A_t, t, s_t)
\tag{5}
$$

B. Network Architecture

As illustrated in Figure 2 (c), the denoising neural network is based on UNet [33] architecture which facilitates multi-scale feature extraction and enhances scalability. Each UNet block is shown in Figure 2 (d) Multi-head self-attention modules are employed in the middle of the network, allowing global receptive field and better capture spatio-temporal dependencies in long and sparse trajectories. The unique design within TrajWeaver denoising neural network is the implementation of state propagation mechanism, which requires solving three key problems: determining the format of states, fusing state s_t into the neural network, and updating s_t to produce s_{t-1}.

Format of State. We design the state as multiple feature sequences, denoted as $s_t = \{f_t^0, f_t^1, f_t^2, ...\}$. This state representation has two advantages. First, since trajectory features are all in sequential format, using the same format for state features can ensure a strong positional correspondence during feature fusing. While other feature formats like vectors do not have this property, it also requires additional processing to fuse vectors with sequences. Second, the dynamic lengths and sparsity levels of trajectories can affect the patterns within

the input contexts at different scale. We make s_t multiple sequences with different scales to align with the UNet's multi-level design, allowing better scalability.

Fusing Previous State. Each feature sequence within s_t is fused into each UNet block as shown in Figure 2 (d). There are several ways for sequence fusing, cross-attention is a widely adopted choice for multi-modal context fusion in diffusion models. However, the multi-scale sequential form of features have already ensure the good positional correspondence, which makes simple addition or concatenation sufficient for state fusion. In the figure, we show state feature fusion using addition.

Propagating State. To update s_t to s_{t-1}, a GRU cell is employed to propagate each f_t^i to f_{t-1}^i. By adopting the GRU cell, s_{t-1} not only encapsulates knowledge in the current denoising step, but also contains knowledge of all previous steps, which allows long-range inter-step information sharing. Ideally, at the last few steps of the denoising process, the state will contain the most useful knowledge extracted in the previous tens to hundreds of steps.

C. Training of TrajWeaver

During TrajWeaver training, a dense ground truth trajectory τ_0 is provided, some time stamps are randomly selected as the query Q, and the corresponding locations are x_0. The diffusion process adds T steps of noise to x_0, producing x_T. Since x_0 is part of τ_0, applying diffusion to x_0 also transforms τ_0 into τ_T.

The training of SPDM differs significantly from typical diffusion model training, because the state propagation introduces more dependency among denoising steps. As suggested in Equation 5 and Figure 2 (b), denoising step t cannot be performed without s_{t+1} from previous step $t+1$. Consequently, unlike in typical diffusion model training where a random t is chosen in each training sample, we must iterate over t from $T-1$ to 0 to ensure that we always have previous state s_{t+1}, with the initial state s_T set to zero tensors.

Another problem is to ensure s_t containing useful features for later steps. Since the s_t is produced by the GRU cell and is used in the next denoising step, the parameters of GRU cannot be optimized when each step is trained independently. Therefore, we have to train multiple steps jointly in each training iteration. The ideal approach is to train the denoising process as a whole, thus the overall objective of SPDM is presented in Equation 6, where θ represents the learnable parameters of the model.

$$
\min_\theta \sum_{t=1}^{T} MSE(\hat{\varepsilon}_{0:t}, \varepsilon_{0:t})
\tag{6}
$$

However, it is unrealistic to train all tens to hundreds of steps in each iteration. Therefore, we split the denoising process into smaller segments consisting of several adjacent steps. The new objective for training 2 steps in a single iteration is formulated in Equation 7.

$$\min_{\theta}(MSE(\hat{\varepsilon}_{0:t+1}, \varepsilon_{0:t+1}) + MSE(\hat{\varepsilon}_{0:t}, \varepsilon_{0:t})) \quad (7)$$

Standard diffusion model samples $\varepsilon_{0:t} \sim \mathcal{N}(0,1)$ independently in each training iteration. This becomes inappropriate when multiple steps are included in one iteration, because $\varepsilon_{0:t+1}$ and $\varepsilon_{0:t}$ are dependent. In fact, $\varepsilon_{0:t+1}$ depends on all previous single-step noises $\{\varepsilon_{0:1}, \varepsilon_{1:2}, ..., \varepsilon_{t:t+1}\}$ and multi-step noises $\{\varepsilon_{0:1}, \varepsilon_{0:2}, ..., \varepsilon_{0:t}\}$. Therefore, $\varepsilon_{0:t}$ and $\varepsilon_{0:t+1}$ cannot be sampled directly from Gaussian distribution. Nonetheless, the single-step noises $\varepsilon_{t:t+1}$ remain independent, which allows us to sample all single-step noises $\{\varepsilon_{t:t+1} | t \in [0, T)\}$ and then derive multi-step noises using Equation IV-C:

First, we combine equation 1 and 2

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

We can also get x_t from x_0 with equation 2, and then replace x_t with a function of x_0 and $\varepsilon_{0:t}$.

$$\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_{0:t+1} = \sqrt{\alpha_t}(\sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\varepsilon_{0:t})$$
$$+ \sqrt{\beta_t}\varepsilon_{t:t+1}$$
$$\sqrt{1-\bar{\alpha}_t}\varepsilon_{0:t+1} = \sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_{t-1}}\varepsilon_{0:t} + \sqrt{\beta_t}\varepsilon_{t:t+1}$$
$$\varepsilon_{0:t+1} = \frac{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_{t-1}}\varepsilon_{0:t} + \sqrt{\beta_t}\varepsilon_{t:t+1}}{\sqrt{1-\bar{\alpha}_t}}$$

The resulting Equation IV-C shows a way to inference $\varepsilon_{0:t+1}$ based on $\varepsilon_{0:t}$ and $\varepsilon_{t:t+1}$. We call it a noise compose equation and simplify it as:
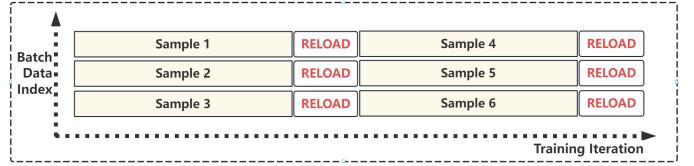
$$\varepsilon_{0:t} \leftarrow Compose(\varepsilon_{0:t+1}, \varepsilon_{t:t+1}) \quad (8)$$

With this equation and randomly sampled list of single-step noises $\{\varepsilon_{0:1}, \varepsilon_{1:2}, ...\varepsilon_{T-1:T}\}$, we can obtain the list of multi-step noises through a recursive process shown below.
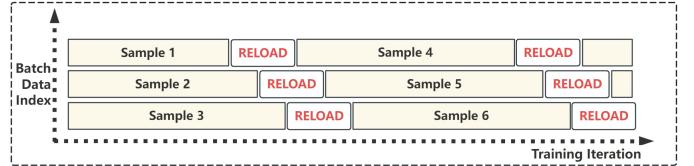
$$\varepsilon_{0:2} \leftarrow Compose(\varepsilon_{0:1}, \varepsilon_{1:2})$$
$$\varepsilon_{0:3} \leftarrow Compose(\varepsilon_{0:2}, \varepsilon_{2:3})$$
$$\varepsilon_{0:4} \leftarrow Compose(\varepsilon_{0:3}, \varepsilon_{3:4})$$
$$...$$
$$\varepsilon_{0:T} \leftarrow Compose(\varepsilon_{0:T-1}, \varepsilon_{T-1:T})$$

Algorithm 1 Training One Sample (2-step)
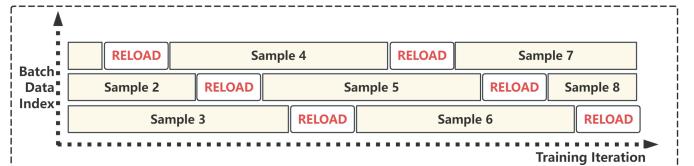
1: Generate $\{\varepsilon_{t:t+1} | t \in [0, T)\}$
2: Derive $\{\varepsilon_{0:t+1} | t \in [0, T)\}$
3: Apply diffusion to τ_0 to get $\{\tau_t | t \in [1, T]\}$
4: $s_T \leftarrow 0$
5: **for** $t \leftarrow T-1$ to 0 **do**
6:     $E_i \leftarrow Embed(C_i)$ for additional contexts.
7:     $A_t, A_{t+1} \leftarrow$ Concat τ_t and τ_{t+1} with τ^l, $\mathbb{M}$, E_i
8:     $\hat{\varepsilon}_{0:t+1}, s_t \leftarrow Net(A_{t+1}, s_{t+1}, t+1)$
9:     $\hat{\varepsilon}_{0:t}, s_{t-1} \leftarrow Net(A_t, s_t, t)$
10:     $loss \leftarrow MSE(\hat{\varepsilon}_{0:t+1}, \varepsilon_{0:t+1}) + MSE(\hat{\varepsilon_{0:t}}, \varepsilon_{0:t})$
11:     Perform Gradient Descent
12: **end for**

(a) Shared t among all samples in a batch, all samples in the batch are reloaded together.

(b) Consecutive t for each sample, the samples in the batch are reloaded one after another.

(c) Uniformly distributed t over the range $[0, T)$, the sample reloading is also uniformly distributed over time.

Fig. 4. Three types of batch managing, the main difference is when to reload each batch sample.

The general principles of SPDM training have been established, which makes TrajWeaver training possible. A naive procedure is presented in Algorithm 1.

While the above design and formulas are sufficient for training, additional techniques are necessary in practical implementation. One major problem is that each trajectory is used T training iterations and the counting down of t applies to the entire batch of trajectories, illustrated as Figure 4a. This shared countdown can lead to over-fitting to certain local range of t and poor generalization across the entire range of t. As a result, the training is very unstable and the convergence extremely slow.

To mitigate this issue, a new batch managing design has to be adopted. We assign a different private t for each sample in a mini-batch, as illustrated in Figure 4b. To further enhance generalization over the entire denoising process and ensure stable convergence, we distribute ts uniformly within the range $[0, T-1]$ as shown in Figure 4c. The t value for each sample is independently updated, and a new sample is loaded in-place when the t for an old sample is reduced to 0.

V. EXPERIMENTS

A. Experimental Settings

Datasets. We use two datasets consisting of dense and smooth taxi trajectories collected in Xi'an city and ChengDu city, China. The third dataset is collected in real-world logistics scenarios generated by couriers on foot, this dataset involves couriers moving in apartment areas and exhibits irregular

Datasets	Logistics	Xi'an	Chengdu
Duration	4 months	1 month	1 month
#Trajectories	2584	131,123	114,110
#Points	759,211	67,134,976	58,424,320

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
	TW-DM	0.017	0.202	2.025
	TW	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
	TW-DM	1.156	3.809	11.062
	TW	1.154	3.597	5.520
JSD ↓ ($\times 10^{-3}$)	DeepMove	1.461	1.786	2.913
	AttnMove	0.661	0.265	2.371
	PriSTI	0.401	0.095	1.488
	DT + RP	1.024	1.846	1.935
	TW-DM	0.253	0.139	0.834
	TW	0.018	0.116	1.084

sample intervals. Table I shows the statistics of the three datasets.

Metrics. We conduct experiments with three evaluation metrics to assess various aspects of seven different methods. They include Mean Squared Error (MSE) between the original trajectory and the recovered trajectory, which aims to measure how effectively models can recover broken trajectories. Normalized Dynamic Time Warping (NDTW) places emphasis on the shape of the trajectory rather than point-to-point comparison. Jensen–Shannon Divergence (JSD) is employed to compare the likelihood of two distributions, serving as a measure of the distribution of points between the recovered trajectories and the trajectories in the original dataset. It is important to note that lower values are preferable for all three metrics.

B. Overall Performance

We conduct comparative experiments among the proposed method and five baselines.

- **DeepMove** [36] is an RNN-based method incorporating an attention mechanism, originally designed for trajectory forecasting. However, due to its reliance on sequential dependencies, it struggles to effectively capture complex relationships among points in trajectory recovery tasks, especially when the trajectories are sparse or irregular.
- **AttnMove** [26] leverages both self-attention and cross-attention mechanisms to learn travel patterns from historical trajectories. It fuses intra- and inter-trajectory

knowledge to enhance trajectory recovery, but may not be fully effective in scenarios where data is highly sparse or contains significant noise.
- **PriSTI** [5] is a diffusion model developed for time-series imputation. We modified it to adapt to trajectory recovery tasks. While PriSTI occasionally achieves performance close to or exceeding TrajWeaver in certain metrics and datasets, it primarily serves as a strong baseline to validate our approach.
- **DT+RP** is a hybrid model that combines DiffTraj [32], a UNet-based diffusion model for generating trajectories, with RePaint [4], a method designed for image inpainting. While DiffTraj generates trajectories without considering the given sparse trajectory, RePaint adapts diffusion-based generation techniques for recovery tasks. This combination aims to bridge the gap between generation and recovery but lacks the specific optimizations present in TrajWeaver.
- **TW** refers to TrajWeaver, the proposed method, which demonstrates superior performance across most metrics and datasets. This result highlights its robustness and effectiveness in recovering both pedestrian and vehicle trajectories in diverse scenarios.
- **TW-DM** is a variant of TrajWeaver that utilizes a standard diffusion model instead of the newly proposed SPDM. While TW-DM achieves competitive results, its performance is slightly inferior to TW, underscoring the benefits of the state propagation mechanism introduced in SPDM.

C. Scalability Analysis

To evaluate the scalability of TrajWeaver across varying trajectory lengths and sparsity levels, we conducted experiments comparing it against strong baselines, including PriSTI and TrajWeaver without the state propagation diffusion model (SPDM). The experiments were performed using the Xi'an dataset, and the results are illustrated in Figure 5. In the three figures on the left, the sparsity level is fixed at 0.5, while the trajectory length is varied from 64 to 512. Conversely, in the three figures on the right, the trajectory length is held constant at 256, while the sparsity level is adjusted from 0.3 to 0.7. This setup allows us to comprehensively assess the model's performance under different conditions of data sparsity and sequence length.

D. Efficiency Analysis

We conducted experiments to demonstrate that TrajWeaver can be both fast and accurate. DDIM is commonly used to accelerate diffusion models, although with some loss in recovery quality. Adding a state propagation pipeline into DDIM introduces 12% more parameters to the model and obtains SP-DDIM. We compared the recovery time and quality of TrajWeaver with DDIM and SP-DDIM. The denoising time was recorded with a batch size of 100 on an NVIDIA GeForce RTX 3070 Ti Laptop GPU. As shown in Table III, the 21-step DDIM trades recovery quality for a 20.7 times
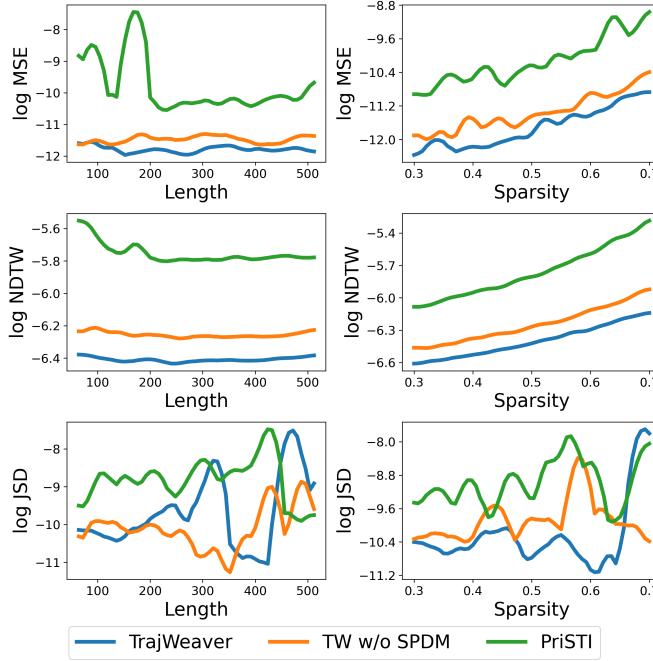
Fig. 5. The comparison among sparsity levels and trajectory lengths on Xi'an dataset.
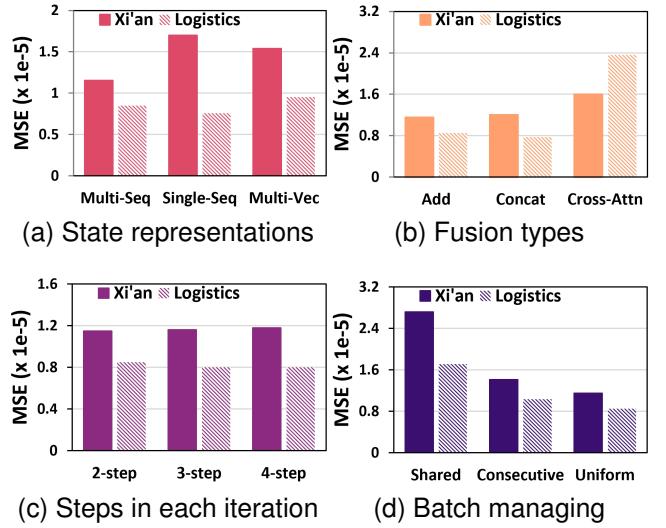

Fig. 6. The comparisons among: (a) Different state representations. (b) State fusion types. (c) Number of denoising steps trained in each training iteration. (d) Three batch managing algorithms introduced in Figure 4.

TABLE III
THE COMPARISON OF RECOVERY TIME AND QUALITY WITH AND WITHOUT STATE PROPAGATION PIPELINE.

Method	Steps	Time (s)	MSE $\times 10^{-5}$	NDTW $\times 10^{-3}$	JSD $\times 10^{-5}$
DDIM	500	65.47	1.78	2.01	3.94
	51	7.31	1.79	2.06	12.58
	21	3.16	2.01	2.31	28.71
SP DDIM	500	80.20	0.86	1.75	3.69
	51	8.58	0.94	1.75	6.78
	21	3.69	1.07	1.92	4.05

speedup compared to the 500-step DDIM. In contrast, 21-step SP-DDIM achieves a 17.7 times speedup which is only 0.53 seconds slower than DDIM with the same number of steps, while still outperforming the 500-step DDIM in recovery quality by approximately 39.89%.

E. Ablation Study

We analyze the effectiveness of different components in SPDM structure and training algorithm using Xi'an and logistics dataset with trajectory length of 512 and sparsity of 0.5. The ablation studies show similar patterns for both dataset, indicating the model can generalize to both pedestrian and vehicle trajectories. Figure 6a shows the outstanding performance of multi-scale sequential state design. Figure 6b indicates that the state fusion can be either addition or concatenation, while cross attention is not very effective. We evaluated training 2, 3 and 4 adjacent steps in one iteration as shown in Figure 6c. While training with more steps is not economic in practice as they cost too much time, computational power and memory. The result indicates that the number of denoising steps trained in a single training iteration has minimal effect

TABLE IV
THE MOVING SPEED AND DISTANCE ESTIMATIONS COMPUTED FROM TRAJECTORIES AFTER NORMALIZATION.

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

on final performance. Finally, Figure 6d proves the robustness of the proposed batch managing algorithm.

F. Case Study

A case study was conducted using trajectories of couriers in a logistics scenario. Monitoring the workload of couriers during last-mile delivery is necessary to ensure they are not overworked and are performing efficiently. Although workload can be measured from blood sugar levels and caloric expenditure, monitoring these indicators requires costly devices and poses privacy issues. Alternatively, accurate moving distance and speed can be used to estimate workload, which relies on dense trajectories. Since collecting and storing dense trajectories consume large amounts of power and storage, sparse trajectories are usually collected and later recovered to dense ones for analysis.

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. TrajWeaver 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
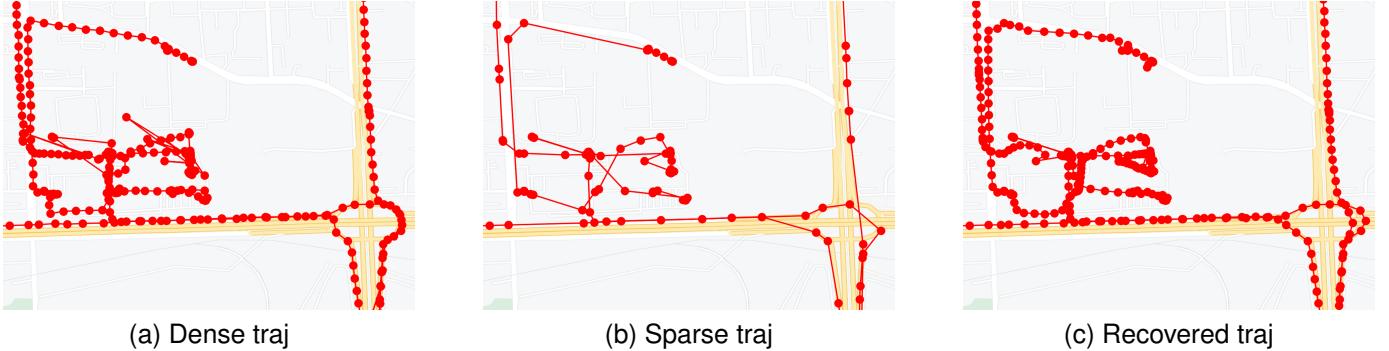
Fig. 7. The dataset of: (a) Dense trajectories as ground truth. (b) Trajectories with the same sparsity as usual logistics scenario. (c) Recovered trajectories using TrajWeaver.

kilometer, and the results are shown in Table IV. 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 7.

VI. CONCLUSION

In conclusion, this paper presents TrajWeaver, a novel diffusion-based framework designed to address the challenges of trajectory recovery in urban environments. TrajWeaver leverages a new variant of the diffusion model, State Propagation Diffusion Model (SPDM), which integrates multiple types of conditions to achieve efficient and high-quality recovery of sparse trajectory data. By introducing a state propagation mechanism, TrajWeaver enhances both the speed and accuracy of the recovery process, ensuring robust performance across different agent types, such as pedestrians and vehicles, even in crowded urban settings. The experimental results demonstrate TrajWeaver's scalability across varying levels of trajectory sparsity and length, and ablation studies confirm the effectiveness of its architectural design and specialized training algorithm. Moreover, the case study illustrates the practical applicability of the proposed method in real-world scenarios, showcasing its potential for deployment in diverse urban analytics tasks.

Looking ahead, there are several promising directions for future research. One potential avenue is to explore the generalization of the SPDM structure beyond trajectory recovery to other domains, such as image or text generation, where efficient and high-quality recovery is also essential. Additionally, further improvements to TrajWeaver could be pursued by incorporating advancements from other diffusion techniques, such as latent diffusion models [13], to enhance performance or reduce computational costs. Future work might also investigate adaptive mechanisms for condition fusion that dynamically adjust to varying levels of data sparsity or explore the integration of real-time data streams to support dynamic, on-the-fly trajectory recovery in rapidly changing urban environments.

REFERENCES

[1] Z. Hong, H. Wang, Y. Ding, G. Wang, T. He, and D. Zhang, "Smallmap: Low-cost community road map sensing with uncertain delivery behavior," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 8, no. 2, may 2024. [Online]. Available: https://doi.org/10.1145/3659596

[2] S. Dabiri, N. Marković, K. Heaslip, and C. K. Reddy, "A deep convolutional neural network based approach for vehicle classification using large-scale gps trajectory data," *Transportation Research Part C: Emerging Technologies*, vol. 116, p. 102644, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X20305593

[3] J. Li, X. Pei, X. Wang, D. Yao, Y. Zhang, and Y. Yue, "Transportation mode identification with gps trajectory data and gis information," *Tsinghua Science and Technology*, vol. 26, no. 4, pp. 403–416, 2021.

[4] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. V. Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," *CoRR*, vol. abs/2201.09865, 2022. [Online]. Available: https://arxiv.org/abs/2201.09865

[5] M. Liu, H. Huang, H. Feng, L. Sun, B. Du, and Y. Fu, "Pristi: A conditional diffusion framework for spatiotemporal imputation," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, Apr. 2023. [Online]. Available: http://dx.doi.org/10.1109/ICDE55515.2023.00150

[6] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.

[7] X. Chen, J. Xu, R. Zhou, W. Chen, J. Fang, and C. Liu, "Trajvae: A variational autoencoder model for trajectory generation," *Neurocomputing*, vol. 428, pp. 332–339, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220312017

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[9] W. Jiang, W. X. Zhao, J. Wang, and J. Jiang, "Continuous trajectory generation based on two-stage gan," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, pp. 4374–4382, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/25557

[10] F. Mazé and F. Ahmed, "Diffusion models beat gans on topology optimization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, 2023, pp. 9108–9116.

[11] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan, and B. Guo, "Vector quantized diffusion model for text-to-image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 696–10 706.

[12] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao, L. He, and L. Sun, "Sora: A review on background, technology, limitations, and opportunities of large vision models," 2024.

[13] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," *CoRR*, vol. abs/2112.10752, 2021. [Online]. Available: https://arxiv.org/abs/2112.10752

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *CoRR*, vol. abs/2006.11239, 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[15] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *CoRR*, vol. abs/2010.02502, 2020. [Online]. Available: https://arxiv.org/abs/2010.02502

[16] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," 2022.

[17] L. Liu, Y. Ren, Z. Lin, and Z. Zhao, "Pseudo numerical methods for diffusion models on manifolds," 2022. [Online]. Available: https://arxiv.org/abs/2202.09778

[18] G. Fang, X. Ma, and X. Wang, "Structural pruning for diffusion models," 2023. [Online]. Available: https://arxiv.org/abs/2305.10924

[19] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park, "One-step diffusion with distribution matching distillation," 2023. [Online]. Available: https://arxiv.org/abs/2311.18828

[20] X. Ma, G. Fang, and X. Wang, "Deepcache: Accelerating diffusion models for free," 2023.

[21] J. Ma, C. Yang, S. Mao, J. Zhang, S. C. Periaswamy, and J. Patton, "Human trajectory completion with transformers," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 3346–3351.

[22] F. Hou, X. Lan, J. Chen, Y. Dong, X. Zhuang, and J. Wu, "A sparse taxi trajectory data recovery and calibrate algorithm," in *2021 China Automation Congress (CAC)*, 2021, pp. 5414–5419.

[23] H. Ren, S. Ruan, Y. Li, J. Bao, C. Meng, R. Li, and Y. Zheng, "Mtrajrec: Map-constrained trajectory recovery via seq2seq multi-task learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1410–1419. [Online]. Available: https://doi.org/10.1145/3447548.3467238

[24] Y. Chen, H. Zhang, W. Sun, and B. Zheng, "Rntrajrec: Road network enhanced trajectory recovery with spatial-temporal transformer," 2022.

[25] J. Wang, N. Wu, X. Lu, W. X. Zhao, and K. Feng, "Deep trajectory recovery with fine-grained calibration using kalman filter," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 3, pp. 921–934, 2021.

[26] T. Xia, Y. Li, Y. Qi, J. Feng, F. Xu, F. Sun, D. Guo, and D. Jin, "History-enhanced and uncertainty-aware trajectory recovery via attentive neural network," *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 3, dec 2023. [Online]. Available: https://doi.org/10.1145/3615660

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[28] J. Si, J. Yang, Y. Xiang, H. Wang, L. Li, R. Zhang, B. Tu, and X. Chen, "Trajbert: Bert-based trajectory recovery with spatial-temporal refinement for implicit sparse trajectories," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4849–4860, 2024.

[29] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[30] Y. Tashiro, J. Song, Y. Song, and S. Ermon, "Csdi: Conditional score-based diffusion models for probabilistic time series imputation," 2021.

[31] J. M. L. Alcaraz and N. Strodthoff, "Diffusion-based time series imputation and forecasting with structured state space models," 2023.

[32] Y. Zhu, Y. Ye, S. Zhang, X. Zhao, and J. J. Q. Yu, "Difftraj: Generating gps trajectory with diffusion probabilistic model," 2023.

[33] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[34] T. Wei, Y. Lin, S. Guo, Y. Lin, Y. Huang, C. Xiang, Y. Bai, M. Ya, and H. Wan, "Diff-rntraj: A structure-aware diffusion model for road network-constrained trajectory generation," 2024. [Online]. Available: https://arxiv.org/abs/2402.07369

[35] Y. He, L. Liu, J. Liu, W. Wu, H. Zhou, and B. Zhuang, "Ptqd: Accurate post-training quantization for diffusion models," 2023. [Online]. Available: https://arxiv.org/abs/2305.10657

[36] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "Deepmove: Predicting human mobility with attentional recurrent networks," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018, p. 1459–1468. [Online]. Available: https://doi.org/10.1145/3178876.3186058