Data-Efficient and Robust Trajectory Generation through Pathlet Dictionary Learning

Yuanbo Tang, Yan Tang, Zixuan Zhang, Zihui Zhao, Yang Li*
Shenzhen Key Laboratory of Ubiquitous Data Enabling,
Tsinghua Shenzhen International Graduate School, Tsinghua University

Trajectory generation has recently drawn growing interest in privacy-preserving urban mobility studies and location-based service applications. Although many studies have used deep learning or generative AI methods to model trajectories and achieved promising results, real-world trajectory data are noisy and often incomplete (e.g., device instability, low sampling rates, privacy-driven partial reporting), introducing distribution shifts and, as observed in our experiments, marked differences between synthetic and real trajectory distributions. To address this issue, we exploit the low-dimensional structure and regular patterns in urban trajectories and propose a parsimonious deep generative model based on sparse pathlet representations, which encode trajectories with sparse binary vectors associated with a learned compact dictionary of trajectory segments. Specifically, we introduce a probabilistic graphical model to describe the trajectory generation process, which includes a Variational Autoencoder (VAE) component and a linear decoder component. During training, the model can simultaneously learn the latent embedding of sparse pathlet representations and the pathlet dictionary that captures essential mobility patterns in the trajectory dataset. The conditional version of our model can also be used to generate customized trajectories based on temporal and spatial constraints. Our model can effectively learn data distribution even using noisy data, achieving relative improvements of 35.4% and 26.3% over strong baselines on two real-world trajectory datasets. Moreover, the generated trajectories can be conveniently utilized for multiple downstream tasks, including trajectory prediction and data denoising. Lastly, the framework design offers a significant efficiency advantage, saving 64.8% of the time and 56.5% of GPU memory compared to previous approaches.

1. Introduction

GPS trajectories contain a wealth of useful information that can be utilized for tasks such as urban planning, traffic management, and location-based services. However, this field also faces several challenges. Firstly, there are significant concerns about privacy leaks. Secondly, in scenarios like newly developed or developing regions, high-quality GPS trajectory data is often scarce due to limited infrastructure. These factors have hindered the development and large-scale application of trajectory-related mining algorithms.

Generative models are a promising solution to these challenges. By synthesizing new datasets that follow the same distribution as the original data, trajectory generative models not only protect data privacy but also facilitates advanced analytical tasks by providing synthetic datasets that reflect real-world behavior and patterns. Besides, customized trajectory generation is also useful for downstream tasks such as navigation and travel route planning. Therefore, this field has garnered widespread interest from researchers in recent years. For instance, Wang et al. proposed a deep learning model called MTNet [1], which considers the topological structure of road networks and road knowledge. Similarly, [2] proposed a model named GDP, which is a diffusion-based model for end-to-end data-driven path planning and trajectory generation that outperforms previous frameworks by incorporating user intentions and road network constraints. However, these deep learning-based methods are typically data-hungry [3] and prone to overfitting when training data is limited.

Despite the promising results achieved in recent research, to the best of our knowledge, few studies have seriously addressed the challenge of noise and data incompleteness in trajectory generative models, which

*Corresponding author: yangli.ai@ieee.org

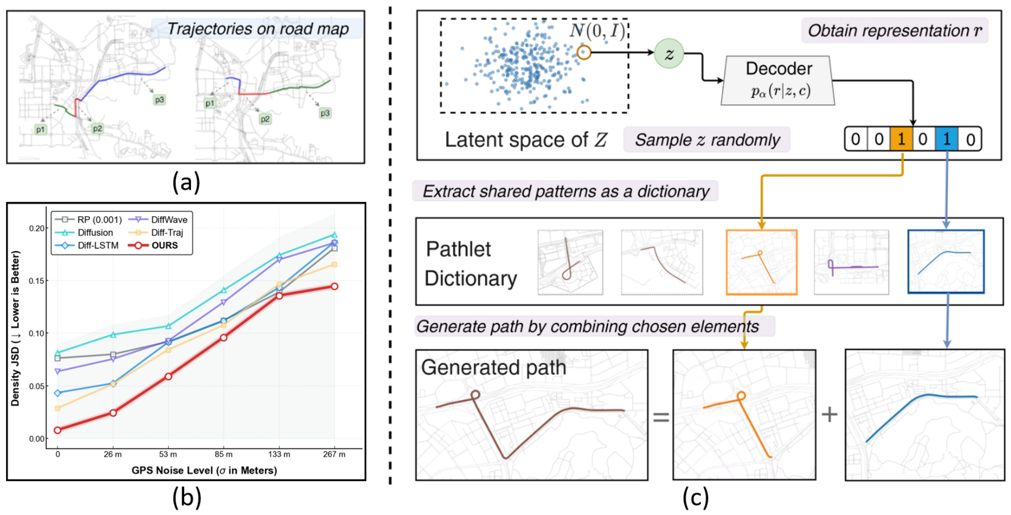

Figure 1: (a) Example of pathlet composition; (b) Comparison of the impact of noise on the distributional fidelity (measured by JSD) of different methods, showing the consistent robustness of our approach; (c) Schematic overview of the framework, supporting both road network-based and grid-based trajectory representations.

are common in real-life situations. Causes of noise and data incompleteness include but are not limited to device instability, environmental interference, low-frequency sampling to reduce storage overhead, and partial data reporting for privacy considerations [4, 5]. Noise and data incompleteness can cause bias in the data distribution, significantly reducing the performance of trajectory generation models (See the Figure 1). While preprocessing methods such as interpolation, filtering and multi-source data fusion can mitigate these issues to some extent before training the generative model [6, 7], these strategies often come with computational overhead. More importantly, preprocessing may also introduce errors that get propagated to the generation stage.

To address these challenges, we propose a **robust** trajectory generation framework, as illustrated in Figure 1. This approach is based on the empirical finding that urban trajectories exhibit strong spatiotemporal regularities [8]. Spatial regularity at a given time can be seen as frequently traveled sub-paths, also known as *pathlets* in [9, 10]. Pathlets serve as the basic units of a trajectory, similar to how words are the basic units of a corpus. Each trajectory can be represented by a binary vector indicating the pathlet index. Given a dictionary of pathlets learned from a large collection of trajectories, we can generate new trajectories by selecting and concatenating elements from the dictionary.

Specifically, we design a probabilistic graphical model to describe the trajectory generation process, implemented using a Variational Autoencoder (VAE) component and a linear decoder component. The VAE is utilized to model the distribution of binary representation vector, while the linear decoder converts the representation vectors to trajectories. During the training process, pathlet dictionary and the VAE model for representation vectors are jointly learned. The pathlet learning objective function aims to find a compact dictionary that reconstructs each trajectory with the least number of elements, resulting in a sparse representation. This enhances robustness and strengthens the ability to retrieve the true underlying signals from noisy data. Moreover, elements in the dictionary often carry semantic meaning. The idea of reconstructing data by combining dictionary elements makes the model's behavior more interpretable.

This work is the first to apply dictionary learning and sparse recovery for trajectory generative models. We validated our approach on two real-world datasets and the experiments revealed that our method generates data more similar to the real data compared to other methods under different levels of noise. Additionally, we further verify the effectiveness of our method in downstream tasks including data recovery and conditional generation.

Overall, our main contributions can be summarized as follows:

- We propose a data-efficient and robust trajectory generation framework that combines the powerful distribution-fitting capabilities of VAE model with the robustness of sparse dictionary learning. Bernoulli sampling is introduced to the VAE reparameterization step to generate binary data.
- We thoroughly evaluated the performance of our method on real-world datasets, and the results show that our method surpasses previous approaches in terms of the Jensen-Shannon Divergence (JSD) score. This validates that our model is more effective in modeling trajectory data distributions in the presence of noise and data scarcity.
- We further validated the utility of the algorithm on two downstream tasks: trajectory denoising and conditional generation. Additionally, we visualized key results and discussed the interpretability of the method. We further analyzed the efficiency of the proposed method.

2. Related work

2.1. Trajectory generative model.

Existing trajectory generation models can be divided into traditional methods and deep learning-based methods. Traditional methods include combining trajectories from the dataset or perturbing the original trajectories to generate new ones [11, 12]. These approaches rely on ad-hoc design, often resulting in datasets with significant distributional differences from the original data, leading to lower usability.

In an effort to solve these problems, recent trends in this field have used deep neural networks to model the intrinsic spatio-temporal distribution of trajectories, with subsequent data generation by sampling from the learned distribution. Researchers formulate raw trajectory data in various formats as inputs for neural networks. Some methods segment GPS sequences into grids, allowing neural networks to learn distribution patterns between grids [13, 14]. However, due to the variability in data distribution across different regions, determining the appropriate grid scale often proves to be a challenging task. Other approaches involve mapping raw GPS data onto road networks and then learning the distribution of edge sequences [15, 16]. The effectiveness of these schemes has been validated in data generation as well as in multiple downstream tasks. Our work, on the other hand, focuses not only on effectiveness but also on robustness and interpretability. Moreover, our framework is flexible enough to support both grid-based and road network-based trajectory representations, making it applicable to trajectory datasets generated in road network environments as well as those collected in free-space scenarios.

2.2. Dictionary learning and sparse recovery.

Our framework design incorporates the concepts of dictionary learning and sparse recovery. Dictionary learning focuses on the training stage with the goal of learning a set of patterns from the dataset. On the other hand, sparse recovery focuses on the application stage, using the learned dictionary to sparsely represent data. Dictionary learning and sparse recovery have been widely applied in various fields [17–19]. Recently, some studies have combined deep neural networks and dictionary learning to leverage the strengths of both. For example, [20] proposed deep convolutional dictionary learning (DCDicL) framework, which adaptively adjusts dictionaries for each image to enhance image restoration, particularly improving denoising performance by preserving subtle structures and textures.

2.3. Dictionary learning for trajectories.

Early work on dictionary learning for trajectory data includes the introduction of the pathlet concept, which was initially proposed by [9] and further developed in subsequent papers [21]. For example, [10] used a randomized rounding algorithm to further optimize the dictionary and explored the effectiveness of trajectory representation across multiple downstream tasks. Previous methods used selection-based strategies for dictionary learning, which were time-consuming. Our approach, based on learning rather than selection strategies, significantly improves efficiency.

3. Preliminary

Terminology. Given a trajectory dataset X, we consider two representation scenarios: (1) *Road network mapping*: trajectories are mapped onto a road network that can be formed as a directed graph $G = (E, V)$, where a trajectory $\boldsymbol{t} \in X$ is defined as a sequence of edges e on G, denoted by $\boldsymbol{t} = \{e_1, e_2, ..., e_n\}$; (2) *Grid-based mapping*: trajectories are represented in a discretized spatial grid where each trajectory $\boldsymbol{t} \in X$ is defined as a sequence of grid cells c, denoted by $\boldsymbol{t} = \{c_1, c_2, ..., c_n\}$. Similarly, a pathlet p refers to a sequence of edges (in road network scenario) or grid cells (in grid-based scenario), and all pathlets together form a dictionary.

Trajectory Vectorization. Each variable-length trajectory $\boldsymbol{t}$ can be transformed into a binary vector $\boldsymbol{x}$ with the size of $|E| \times 1$, where $|E|$ refers to the number of spatial units (edges in road network mapping or grid cells in grid-based mapping). Each entry $\boldsymbol{x}_i = 1$ if the corresponding spatial unit is covered by this trajectory, and $\boldsymbol{x}_i = 0$ otherwise. Similarly, the binary vector $\boldsymbol{r}$ is used to store which elements from the dictionary D are selected for reconstruction.

Problem statement (Trajectory Generation). Given a trajectory dataset X represented either through road network mapping or grid-based discretization, the goal of the trajectory generation task is to learn a generative model M using X, which can synthesize trajectory data Y that satisfies the following:

- *Authenticity*: The generated trajectories Y retain the spatiotemporal properties and distribution of the original trajectory dataset X.
- *Utility*: The generated trajectories and the generative model M can benefit various downstream tasks.

Besides the above two criteria that are common to any generative models, an ideal model M should also have:

- *Robustness and Interpretability*: The algorithm can effectively learn data distribution from noisy or incomplete dataset and its internal structure, behavior, and outcomes are easy to understand.

4. Methodology

4.1. Generative Model

In this work, we formulate the compositional trajectory generation problem using the following Markov chain: $\boldsymbol{z} \rightarrow \boldsymbol{r} \rightarrow \boldsymbol{x}$. Let $\boldsymbol{x} \in \{0,1\}^{|E|}$ denote the observed trajectory on a road network with edge set E, and $\boldsymbol{r} \in \{0,1\}^n$ be its sparse binary representation over a pathlet dictionary of size n. We introduce a K-dimensional latent Gaussian variable $\boldsymbol{z} \in \mathbb{R}^K$ that generates the representation $\boldsymbol{r}$.

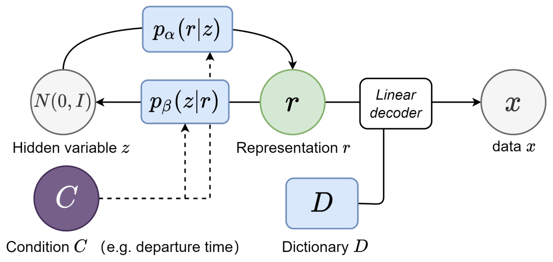

Figure 2: Illustration of the generative framework which describes the generative process of path.

To enable gradient-based optimization, we adopt a continuous relaxation of the binary trajectory and assume a Gaussian observation model:

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, I_K), \qquad \boldsymbol{x} \mid \boldsymbol{r} \sim \mathcal{N}(D\boldsymbol{r}, \sigma^2 I_{|E|}), \tag{1}$$

where $D \in \{0,1\}^{|E| \times n}$ is the pathlet dictionary matrix and σ^2 is a fixed noise variance. Although $\boldsymbol{x}$ is represented as a binary vector in $\{0,1\}^{|E|}$, the Gaussian observation model serves as a continuous approximation that facilitates differentiable training.

The dependency between $\boldsymbol{z}$ and $\boldsymbol{r}$ is modeled using a Binary VAE architecture [22]. Given $\boldsymbol{z}$, two neural networks f_{α^m} and f_{α^p} produce the parameters $\boldsymbol{m}$ and $\boldsymbol{p}$ of a hierarchical Bernoulli distribution:

$$\boldsymbol{m} = \exp\big(f_{\alpha^m}(\boldsymbol{z})\big), \qquad \boldsymbol{p} = \text{sigmoid}\big(f_{\alpha^p}(\boldsymbol{z})\big), \tag{2}$$

and each dimension of $\boldsymbol{r}$ is generated as

$$r_j \mid \boldsymbol{z} \sim \text{Bernoulli}\big(1 - p_j^{m_j}\big), \quad j = 1, \dots, n, \tag{3}$$

where p_j and m_j denote the j-th components of $\boldsymbol{p}$ and $\boldsymbol{m}$, respectively, and $p_j^{m_j}$ denotes element-wise exponentiation.

Under this construction, the joint distribution of all variables factorizes as

$$p(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{r}; D)\, p_\alpha(\boldsymbol{r} \mid \boldsymbol{z})\, p(\boldsymbol{z}), \tag{4}$$

where $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, I_K)$ and $p_\alpha(\boldsymbol{r} \mid \boldsymbol{z})$ is induced by the Bernoulli model above.

4.2. Variational learning

From the joint distribution in Eq. (4), the negative log-density can be decomposed as

$$-\log p(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{z}) = -\log p(\boldsymbol{x} \mid \boldsymbol{r}; D) - \log p_\alpha(\boldsymbol{r} \mid \boldsymbol{z}) - \log p(\boldsymbol{z}). \tag{5}$$

Since $p(\boldsymbol{r}, \boldsymbol{z}) = p_\alpha(\boldsymbol{r} \mid \boldsymbol{z}) p(\boldsymbol{z})$, the last two terms can be grouped together, yielding

$$-\log p(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{z}) = -\log p(\boldsymbol{x} \mid \boldsymbol{r}; D) - \log p(\boldsymbol{r}, \boldsymbol{z}). \tag{6}$$

This decomposition shows that the overall objective naturally splits into two complementary parts: the first term $-\log p(\boldsymbol{x} \mid \boldsymbol{r}; D)$ is captured by the MDL-based dictionary learning objective in Section 4.3, which characterizes the generative process of trajectories conditioned on $\boldsymbol{r}$; the second term $-\log p(\boldsymbol{r}, \boldsymbol{z})$ is modeled by the Binary VAE that learns a prior over the representation vectors, as described in this subsection.

Given a collection of representation vectors, our goal is to learn the generative model $p_\alpha(\boldsymbol{r} \mid \boldsymbol{z}) p(\boldsymbol{z})$ by maximizing the marginal log-likelihood $\sum_i \log p(\boldsymbol{r}^{(i)})$, where

$$\log p(\boldsymbol{r}) = \log \int p_\alpha(\boldsymbol{r} \mid \boldsymbol{z})\, p(\boldsymbol{z})\, d\boldsymbol{z}. \tag{7}$$

The integral over $\boldsymbol{z}$ is intractable for a flexible neural decoder, so we resort to variational inference. We introduce an encoder network g_ϕ that parameterizes a variational posterior

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{r}) = \mathcal{N}\big(\boldsymbol{\mu}_\phi(\boldsymbol{r}), \text{diag}(\boldsymbol{\sigma}_\phi^2(\boldsymbol{r}))\big), \tag{8}$$

and use the reparameterization trick to draw samples from $q_\phi(\boldsymbol{z} \mid \boldsymbol{r})$.

The evidence lower bound (ELBO) for $\log p(\boldsymbol{r})$ is given by

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{r}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{r})}\left[\log p_\alpha\left(\boldsymbol{r} \mid \boldsymbol{z}\right)\right] - \text{KL}[q_\phi\left(\boldsymbol{z} \mid \boldsymbol{r}\right) \| \, p\left(\boldsymbol{z}\right)], \tag{9}$$

which satisfies $\mathcal{L}_{\text{ELBO}}(\boldsymbol{r}) \leq \log p(\boldsymbol{r})$. For our Bernoulli decoder, the reconstruction term can be written as

$$\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{r})}\left[\log p_\alpha\left(\boldsymbol{r} \mid \boldsymbol{z}\right)\right] = \sum_{j=1}^{n} \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{r})}\left[\boldsymbol{r}_j \log\left(1 - \boldsymbol{p}_j^{\boldsymbol{m}}\right) + (1 - \boldsymbol{r}_j) \log\left(\boldsymbol{p}_j^{\boldsymbol{m}}\right)\right], \tag{10}$$

where $\boldsymbol{m}$ and $\boldsymbol{p}$ are functions of $\boldsymbol{z}$ as defined in Eq. (2). The second component is the Kullback–Leibler (KL) divergence, which measures the discrepancy between the learned posterior $q_\phi\left(\boldsymbol{z} \mid \boldsymbol{r}\right)$ and the prior distribution $p\left(\boldsymbol{z}\right) = \mathcal{N}(\boldsymbol{0}, I_K)$.

In practice, we minimize the negative ELBO as the Binary VAE loss:

$$L_{\text{VAE}} = -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{r})}\left[\log p_\alpha\left(\boldsymbol{r} \mid \boldsymbol{z}\right)\right] + \text{KL}[q_\phi\left(\boldsymbol{z} \mid \boldsymbol{r}\right) \| \, p\left(\boldsymbol{z}\right)], \tag{11}$$

which encourages the model to accurately reconstruct $\boldsymbol{r}$ while keeping the latent representation $\boldsymbol{z}$ close to the Gaussian prior.

4.3. MDL-based dictionary learning

We use a compositional learning based approach to automatically learn the pathlet dictionary and representation from dataset. Based on the Minimum Description Length (MDL) principle [23], we extend the original reconstruction error term $-\log p(\boldsymbol{x}|\boldsymbol{r}; D)$ to include three components: (1) the reconstruction error, (2) a dictionary complexity regularization term, and (3) a representation sparsity regularization term. The latter two terms are derived from the MDL principle as follows.

In the MDL framework, the total description length for a data set is decomposed into the length for encoding the model $L(H)$ and the length for describing the data given the model $L(D|H)$. Here, the matrix D serves as the model (dictionary), and R is its corresponding sparse data representation. A visualization of the relationship between these matrices D, X, and R can be found in Figure 4 in Section B of the Appendix.

Model Length: Each dictionary atom corresponds to a column of $D \in \{0,1\}^{|E|\times n}$, where the binary entries encode the inclusion of graph edges (or grids) in each atom. Since D is binary, storing one atom requires $|E|$ bits. The total storage cost of the effective part of the dictionary (i.e., atoms utilized in the representation) is then: $L(H) = |E| \cdot |\mathcal{A}_{\text{eff}}|$, where $|\mathcal{A}_{\text{eff}}|$ denotes the number of dictionary atoms that are actually used.

The effective dictionary size is determined by examining the usage matrix R. An atom $D^{(j)}$ (where $D^{(j)}$ denotes the j-th column of matrix D) is effective if its corresponding row R_j (where R_j denotes the j-th row of matrix R) contains any nonzero element: $\mathbf{1}_{\{D^{(j)} \text{ is effective}\}} = \mathbf{1}_{\{\max_i R_{ji} > 0\}}$, where R_{ji} denotes the (j,i)-th element of matrix R, and $\mathbf{1}_{\{condition\}}$ denotes the indicator function, which equals 1 if the condition holds and 0 otherwise. For binary R, the effective dictionary size becomes:

$$|\mathcal{A}_{\text{eff}}| = \sum_{j=1}^{n} \mathbf{1}_{\{\max_i R_{ji} > 0\}} = \sum_{j=1}^{n} \max_i R_{ji} \tag{12}$$

Data Representation Length: Given a fixed D, the representation R is a binary sparse matrix. The length to encode R is determined by the number of nonzero elements, therefore $L(D|H) = \|R\|_0$. And for binary R, the L_0 pseudo-norm coincides with the L_1 norm. In summary, the MDL for this dictionary learning setting takes the form

$$\text{MDL} = L(H) + L(D|H) = |E| \cdot |\mathcal{A}_{\text{eff}}| + \|R\|_1 = |E| \cdot \sum_{j=1}^{m} \max_i R_{ji} + \|R\|_1 \tag{13}$$

For the reconstruction error term, given our assumption that $\boldsymbol{x} = D\boldsymbol{r} + \epsilon$ where ϵ represents Gaussian noise, we have: $P(\boldsymbol{x}|\boldsymbol{r}; D) = \mathcal{N}(\boldsymbol{x}; D\boldsymbol{r}, I)$. Maximizing this probability is equivalent to minimizing $\|\boldsymbol{x} - D\boldsymbol{r}\|_2^2$.

Combining the reconstruction error with the MDL-derived regularization terms, the complete compositional learning loss function is defined as:

$$L_{\text{dict}} = \|X - DR\|_2^2 + \lambda_1 \sum_{j=1}^{m} \max_i R_{ji} + \lambda_2 \|R\|_1 \tag{14}$$

where λ_1 and λ_2 are hyperparameters that control the trade-off between reconstruction accuracy and model complexity. The first term ensures faithful reconstruction of the original paths, while the second and third terms encourage dictionary compactness and representation sparsity, respectively. This loss function L_{dict} is consistent with the one proposed in [10]. However, our work provides a new interpretation based on the MDL principle, which offers a principled theoretical foundation for understanding why this particular combination of terms leads to effective dictionary learning.

Problem formulation: Combining both components, our complete objective function is:

$$\min_{\alpha,\beta,D,R} L_{\text{VAE}} + L_{\text{dict}}, \quad \text{s.t.} \quad D_{i,j} \in \{0,1\},\ R_{i,j} \in \{0,1\} \tag{15}$$

4.4. Training Procedure

To solve the aforementioned optimization problem, we have designed a two-step algorithm. Since neural networks require differentiable operations for gradient-based optimization, we first relax the integer optimization constraints to interval constraints $[0, 1]$, then obtain a decimal solution using projected gradient descent, and finally convert it into a binary solution through probabilistic rounding method.The detailed procedures are provided in Algorithm 1 in the Appendix B.

Our approach essentially involves learning the distribution of a given dataset and then sampling from it to generate new paths. Therefore, an unavoidable issue is that the connectivity of the generated paths is not strictly guaranteed. To address this problem, we design specific post-processing algorithm with detailed procedures provided in Algorithm 2 in the Appendix B.

4.5. Downstream tasks

Once the generative model has been obtained, it captures the distribution of the data. Our proposed generative model can be conveniently applied to various downstream tasks, demonstrating its versatility and practical value. We focus on two key applications: conditional path generation and data denoising with noisy paths.

The conditional generation capability allows users to specify certain conditions or attributes to generate data that meets specific requirements, which can be achieved by replacing the VAE with a CVAE. Figure 5 illustrates the architecture of conditional Pathlet-VAE used in our framework. Additionally, our framework's inherent sparsity and dictionary-based design make it naturally robust to noise, enabling effective data denoising applications. For detailed methodologies, experimental setups, and comprehensive results of these downstream tasks, please refer to Appendix B.

5. Experiment

In this section, we conduct extensive experiments to comprehensively evaluate the proposed framework from four aspects: effectiveness, efficiency, robustness, and interpretability, and compare it with previous methods. We first briefly introduce the datasets, experimental setup, and evaluation protocol. After that, we will attempt to answer the following research questions:

RQ1: Is our proposed generative model capable of learning the true data distribution and generating datasets that closely approximate the true distribution? **RQ2:** Can our algorithm be used on noisy or incomplete data? To what extent of noise can it operate normally? **RQ3:** Can the model be conveniently applied to different downstream tasks while achieving good performance? **RQ4:** What does the representation vector generated signify? Can it be visualized or understood in an intuitive way? **RQ5:** How efficient is the proposed method?

5.1. Experiment setup

We conduct extensive experiments on multiple real-world taxi trajectory datasets. To validate robustness, we simulate various noise levels by randomly replacing trajectory nodes with "unknown" nodes following a Bernoulli distribution. Detailed experimental setup, including dataset descriptions, data preprocessing steps, noise simulation protocols, and baseline model configurations, are provided in Appendix C.

5.2. Numerical Comparison with Previous Work

Evaluation Protocol and Baselines: We use Jensen-Shannon Divergence (JSD) to quantitatively measure the distribution distance between generated and real trajectory datasets. We compare our method against multiple state-of-the-art baselines across both road network-based and grid-based representations. Detailed evaluation metrics, baseline descriptions are provided in Appendix C.4.

Numerical Results: Table 1 presents a comprehensive numerical comparison of our approach with other methods under different noise levels across both road network-based and grid-based representations. Results show that in the absence of noise, the datasets generated by various methods exhibit similar performance. As the noise level increases, all methods produce datasets that deviate further from the true distribution. However, our method demonstrates superior robustness across both representation types.

For road network-based representation, traditional methods like MTnet and GDP show significant performance degradation as noise increases. On the Shenzhen dataset, when the noise level exceeds 0.04, our

Table 1: The performance comparison with previous work across varying noise levels. Lower is better. The best results are highlighted in **bold**, while the second-best results are underlined.

Road Network-based Representation												
Dataset	ShenZhen						Porto					
Noise level	0	0.02	0.04	0.06	0.08	0.10	0	0.02	0.04	0.06	0.08	0.10
Noisy data	0.00	0.04	0.09	0.12	0.15	0.17	0.00	0.05	0.10	0.14	0.18	0.22
MTnet [1]	0.08	0.20	0.23	0.24	0.25	0.28	0.14	0.21	0.28	0.28	0.29	0.29
GDP [2]	0.09	0.14	0.17	0.18	0.18	0.19	**0.07**	0.17	0.18	0.15	0.18	0.21
Binary VAE	0.10	0.13	0.15	0.16	0.18	0.20	0.13	0.17	0.19	0.22	0.27	0.27
L1B-VAE	0.07	0.11	0.13	0.17	0.18	0.20	0.10	0.16	0.19	0.23	0.26	0.27
Our method	**0.07**	**0.07**	**0.09**	**0.11**	**0.13**	**0.15**	0.11	**0.11**	**0.13**	**0.15**	**0.16**	**0.18**

Grid-based Representation												
Dataset	ShenZhen						Porto					
Noise level	0	2.5e-4	5e-4	8e-4	1.25e-3	2.5e-3	0	2.5e-4	5e-4	8e-4	1.25e-3	2.5e-3
Noisy data	0.00	0.02	0.06	0.09	0.13	0.18	0.00	0.03	0.09	0.13	0.17	0.20
Diffusion	0.08	0.10	0.11	0.14	0.17	0.19	0.13	0.16	0.17	0.20	0.21	0.22
diff-lstm [24]	0.04	0.05	0.09	0.11	0.14	0.18	0.12	0.15	0.16	0.17	0.18	0.22
diffwave [25]	0.06	0.08	0.09	0.13	0.17	0.19	0.12	0.13	0.15	0.17	0.19	0.21
diff-traj[24]	0.03	0.05	0.08	0.11	0.15	0.17	0.08	0.10	0.13	0.15	0.18	0.21
Our method	**0.01**	**0.02**	**0.06**	**0.10**	**0.14**	**0.14**	**0.02**	**0.04**	**0.10**	**0.14**	**0.17**	**0.20**

Table 2: Performance comparison of data efficiency.

	ShenZhen			Porto		
Method	5%	20%	80%	5%	20%	80%
Diffusion	0.19	0.07	0.05	0.25	0.16	0.06
diff-lstm	0.16	0.12	0.13	0.17	0.11	0.05
diffwave	0.18	0.07	0.04	0.18	0.08	0.05
diff-traj	0.18	0.06	0.04	0.17	0.07	0.05
Ours	**0.01**	**0.01**	**0.01**	**0.06**	**0.04**	**0.04**

Table 3: Next-edge prediction performance on the Shenzhen and Porto vehicle trajectory datasets. We report the average cross-entropy loss (lower is better) over all time steps in the test set for each city.

Method	Shenzhen ↓	Porto ↓	Avg ↓
Uniform (outgoing)	1.51	1.53	1.52
First-order Markov	1.11	1.20	1.15
LSTM-Edge	0.94	0.98	0.96
MTnet [1]	0.90	0.95	0.93
Our method	**0.81**	**0.88**	**0.84**

method generates datasets closer to the true distribution than even the noise-affected original data. For grid-based representation, the performance improvements are even more pronounced. Diffusion-based methods show substantial degradation under noise. Notably, our method achieves the best performance across all noise levels on both datasets, with particularly significant improvements on the Porto dataset where the performance gap widens as noise increases. *These comprehensive experiments across both representation types demonstrate that our method can effectively learn robust data distributions and significantly outperforms existing approaches under noisy conditions.* (For **RQ1-2**)

Table 2 demonstrates the superior data efficiency of our approach. When training data is limited to 5% of the full dataset, our method achieves JSD values of 0.01 on Shenzhen and 0.06 on Porto, significantly outperforming baseline methods. Our method maintains consistent performance across all data sizes (5%-80%), while baseline methods exhibit substantial performance fluctuations. This data efficiency stems from the dictionary learning framework's ability to capture essential mobility patterns with limited samples, making it particularly valuable for real-world applications where trajectory data collection is expensive or privacy-constrained.

5.3. Application in trajectory prediction

Problem setup (next-edge prediction). Given a trajectory on the road network $\boldsymbol{t} = (e_1, e_2, \ldots, e_L)$, we construct supervised samples from each prefix: at step t ($1 \leq t < L$), the input consists of the prefix $(e_1, \ldots, e_t)$ and the target label is the next edge $y_t = e_{t+1}$. Let $\mathcal{E}$ denote the set of edges in the road network and $p_\theta(e \mid e_{1:t})$ be the model-predicted probability of choosing edge e as the next step. We evaluate models by the average cross-entropy loss over all time steps in the test set: $\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta\big(y_i \mid \text{prefix}_i\big)$, where $y_i \in \mathcal{E}$ is the ground-truth next edge for the i-th sample and N is the total number of prediction instances.

Baselines. We compare our method with several lightweight next-edge predictors, including a uniform outgoing-edge model, a first-order Markov model, and an LSTM-based sequence model. Our model uses the learned pathlet-based representation to produce next-edge probabilities under the same setting.

Table 3 shows that while probabilistic baselines and the LSTM-Edge model already capture useful local transition patterns, our method consistently achieves the lowest cross-entropy on both Shenzhen and Porto. This indicates that the pathlet-based representation learned by our generative model provides more informative features for downstream prediction.

Appendix D further presents a conditional trajectory prediction case study and data denoising results, where our method achieves up to 68% noise reduction on the Porto dataset. *These downstream experiments demonstrate the utility of our method.* (For **RQ3**)

5.4. Interpretability

Besides robustness, incorporating pathlet dictionary learning into the architecture of the generative model offers the additional benefit of enhanced interpretability, which is crucial for trust and usability. Figure 3(a) illustrates the dictionary elements learned by the model, showing meaningful mobility patterns that capture commonly used routes. For example, the third row shows a pathlet in Porto, which represents a route crossing from the north bank to the south bank of the Douro River via the Dom Luís I Bridge. This is an important traffic route. Figure 3(b) demonstrates how trajectory samples are concatenated and generated using elements from the dictionary. *The visualization of critical parameters and outcomes within the model assists us in gaining a deeper understanding of the model's behavior.* (For **RQ4**)

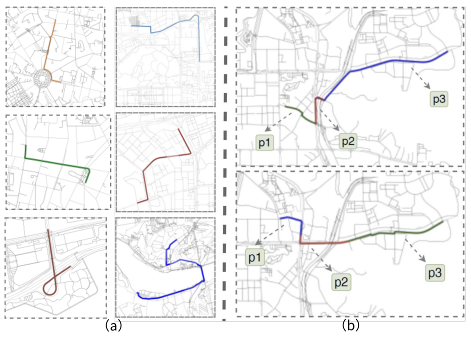

Figure 3: Visualization of pathlets and trajectory decomposition.

5.5. Efficiency Analysis

One major challenge in obtaining a pathlet dictionary from trajectory data is that the space of candidate dictionary elements grows exponentially with the scale of the road network, suffering from the curse of dimensionality. [10] use pre-filtering and hierarchical learning strategies to keep the computational cost within an acceptable range, albeit by sacrificing some performance. [9] employed a dynamic programming approach, which could not leverage the computational acceleration benefits of modern GPUs. (The two aforementioned works are respectively referred to as GD and DP in Table 4.)

Table 4: The efficiency comparison of different methods.

Method	Dictionary Size	Representation Cost	GPU Memory	Running Time
DP	2.16k	2.13	-	2.2h
GD	1.67k	2.05	30.1G	0.85h
Ours	1.03k	1.96	13.1G	0.3h

In contrast to prior research, our work is the first to adopt a learning-based rather than a selection-based strategy, which provides significant efficiency advantages to our method. *It is evident that our scheme not only surpasses the previous ones in terms of dictionary size and expression cost, but also reduces training time by 64.8% and 86.4%, respectively, and cuts down GPU memory usage by 56.5%.* (For **RQ5**)

6. Conclusion

In this paper, we introduce a novel framework that integrates VAE and sparse pathlet dictionary learning for trajectory generation. Through experiments on multiple datasets, we found that the learned data distribution under various levels of noise is closer to the real distribution compared to benchmark models. Furthermore, we investigated its performance on downstream tasks and visualized the dictionary learning process, demonstrating that our framework offers more interpretability compared to purely deep neural networks.

In future work, we will delve deeper into the theoretical principles and guarantees of its robustness, as well as explore its application in a wider range of scenarios.

References

[1] Yong Wang, Guoliang Li, Kaiyu Li, and Haitao Yuan. A Deep Generative Model for Trajectory Modeling and Utilization. *Proceedings of the VLDB Endowment*, 16(4):973–985, December 2022. ISSN 2150-8097. doi: 10.14778/3574245.3574277.

[2] Dingyuan Shi, Yongxin Tong, Zimu Zhou, Ke Xu, Zheng Wang, and Jieping Ye. GRAPH-CONSTRAINED DIFFUSION FOR END-TO-END PATH PLANNING. 2024.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[4] Ziqiao Liu, Hao Miao, Yan Zhao, Chenxi Liu, Kai Zheng, and Huan Li. LightTR: A Lightweight Framework for Federated Trajectory Recovery, May 2024.

[5] Yang Li, Yangyan Li, Dimitrios Gunopulos, and Leonidas Guibas. Knowledge-based trajectory completion from sparse GPS samples. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPACIAL '16, pages 1–10, New York, NY, USA, October 2016. Association for Computing Machinery. ISBN 978-1-4503-4589-7. doi: 10.1145/2996913.2996924.

[6] Huimin Ren, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li, and Yu Zheng. MTrajRec: Map-Constrained Trajectory Recovery via Seq2Seq Multi-task Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 1410–1419, New York, NY, USA, August 2021. Association for Computing Machinery. ISBN 978-1-4503-8332-5. doi: 10.1145/3447548.3467238.

[7] YinYifang, ShahRajiv Ratn, WangGuanfeng, and ZimmermannRoger. Feature-based Map Matching for Low-Sampling-Rate GPS Trajectories. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, August 2018. doi: 10.1145/3223049.

[8] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *nature*, 453(7196):779–782, 2008.

[9] Chen Chen, Hao Su, Qixing Huang, Lin Zhang, and Leonidas Guibas. Pathlet learning for compressing and planning trajectories. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 392–395, Orlando Florida, November 2013. ACM. ISBN 978-1-4503-3521-9. doi: 10.1145/2525314.2525443.

[10] Yuanbo Tang, Zhiyuan Peng, and Yang Li. Explainable Trajectory Representation through Dictionary Learning, December 2023.

[11] Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. Towards trajectory anonymization: A generalization-based approach. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '08, pages 52–61, New York, NY, USA, November 2008. Association for Computing Machinery. ISBN 978-1-60558-324-2. doi: 10.1145/1503402.1503413.

[12] Paul A. Zandbergen. Ensuring Confidentiality of Geocoded Health Data: Assessing Geographic Masking Strategies for Individual-Level Data. *Advances in Medicine*, 2014(1):567049, 2014. ISSN 2314-758X. doi: 10.1155/2014/567049.

[13] Xi He, Graham Cormode, Ashwin Machanavajjhala, Cecilia Procopiuc, and Divesh Srivastava. DPT : Differentially private trajectory synthesis using hierarchical reference systems. In *Proceedings of the VLDB Endowment*, volume 8, pages 1154–1165, Hawaii, 2015.

[14] Xiangjie Kong, Feng Xia, Zhaolong Ning, Azizur Rahim, Yinqiong Cai, Zhiqiang Gao, and Jianhua Ma. Mobility Dataset Generation for Vehicular Social Networks Based on Floating Car Data. *IEEE Transactions on Vehicular Technology*, 67(5):3874–3886, May 2018. ISSN 1939-9359. doi: 10.1109/TVT.2017.2788441.

[15] Seongjin Choi, Jiwon Kim, and Hwasoo Yeo. TrajGAIL: Generating urban vehicle trajectories using generative adversarial imitation learning. *Transportation Research Part C: Emerging Technologies*, 128: 103091, July 2021. ISSN 0968-090X. doi: 10.1016/j.trc.2021.103091.

[16] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. Modeling Trajectories with Recurrent Neural Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3083–3090, Melbourne, Australia, August 2017. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3. doi: 10.24963/ijcai.2017/430.

[17] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, 19(11):2861–2873, November 2010. ISSN 1941-0042. doi: 10.1109/TIP.2010.2050625.

[18] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Yi Ma. Robust Face Recognition via Sparse Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(31):210–227, 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.79.

[19] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse Representation for Color Image Restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, January 2008. ISSN 1941-0042. doi: 10.1109/TIP.2007.911828.

[20] Hongyi Zheng, Hongwei Yong, and Lei Zhang. Deep convolutional dictionary learning for image denoising. pages 630–641, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Zheng_Deep_Convolutional_Dictionary_Learning_for_Image_Denoising_CVPR_2021_paper.html.

[21] Gian Alix and Manos Papagelis. PathletRL: Trajectory Pathlet Dictionary Construction using Reinforcement Learning. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '23, pages 1–12, New York, NY, USA, December 2023. Association for Computing Machinery. ISBN 9798400701689. doi: 10.1145/3589132.3625622.

[22] He Zhao, Piyush Rai, Lan Du, Wray Buntine, Dinh Phung, and Mingyuan Zhou. Variational Autoencoders for Sparse and Overdispersed Discrete Data. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 1684–1694. PMLR, June 2020.

[23] Mark H. Hansen and Bin Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001. doi: 10.1198/016214501753168398.

[24] Yuanshao Zhu, Yongchao Ye, Shiyao Zhang, Xiangyu Zhao, and James J Q Yu. DiffTraj: Generating GPS Trajectory with Diffusion Probabilistic Model. 2023.

[25] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *ICLR*, 2021. URL https://iclr.cc/virtual/2021/poster/2979.

[26] Desheng Zhang, Juanjuan Zhao, Fan Zhang, and Tian He. UrbanCPS: A cyber-physical system based on multi-source big infrastructure data for heterogeneous model integration. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, ICCPS '15, pages 238–247, New York, NY, USA, April 2015. Association for Computing Machinery. ISBN 978-1-4503-3455-6. doi: 10.1145/2735960.2735985.

[27] https://people.cs.rutgers.edu/~dz220/Data.html, November 2022.

[28] Luis Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. Predicting Taxi–Passenger Demand Using Streaming Data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402, September 2013. ISSN 1558-0016. doi: 10.1109/TITS.2013.2262376.

[29] Wannes Meert and Mathias Verbeke. HMM with non-emitting states for Map Matching. In *European Conference on Data Analysis (ECDA)*, page 1, 2018.

[30] Zihui Zhao, Yuanbo Tang, Jieyu Ren, Xiaoping Zhang, and Yang Li. A unified probabilistic framework for dictionary learning with parsimonious activation. *ArXiv*, abs/2509.25690, 2025. URL https://api.semanticscholar.org/CorpusID:281682122.

A. Appendix

This appendix provides comprehensive supplementary material for the Pathlet Variational Auto-Encoder framework. We present: (1) method details including conditional path generation with CVAE architecture, data denoising capabilities, and a greedy post-processing algorithm for trajectory connectivity; (2) detailed experimental setups covering two datasets (Shenzhen, Porto) with preprocessing procedures, noise simulation protocols, JSD evaluation metrics, and five baseline comparisons; (3) additional experimental results demonstrating data denoising performance; and (4) complete mathematical notation table. These materials provide essential technical details supporting the effectiveness and interpretability of our approach.

B. Matrix Visualization

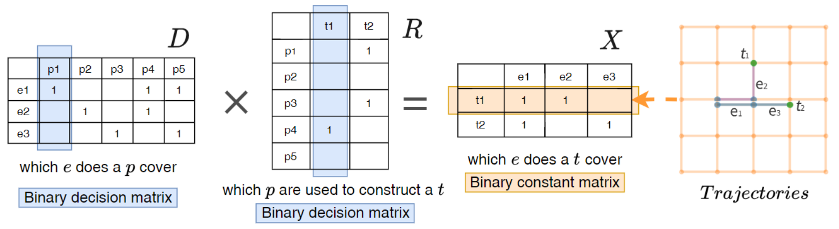

Figure 4: Illustration of the matrices D, X, R (visualized in edge form): X is the trajectory matrix generated from the dataset; D refers to the pathlet dictionary matrix; R is the representation matrix where each column corresponds to a representation vector.

C. Method Details

C.1. Conditional Path Generation

Compared to unconditional generation, conditional generation allows the specification of certain conditions or attributes, thereby generating data that meets specific requirements. This control capability is particularly useful when data with specific characteristics is needed. This capability can be achieved in our framework with a simple modification: replacing the VAE with a CVAE. The figure below illustrates a specific scenario where, for example, the first half and departure time of the path are provided as conditions along with the input, and the model generates the complete path. For the encoding method, we map the departure time to 24 hourly intervals and use one-hot encoding. In this case, the framework can perform the path prediction task.

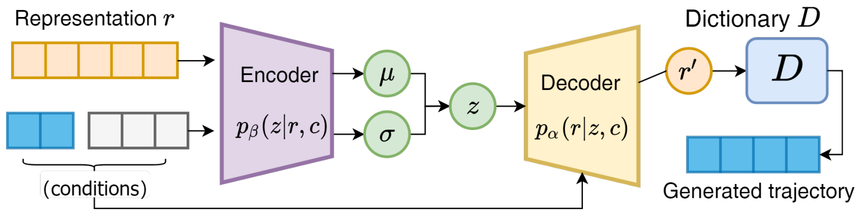

Figure 5: Architecture of Conditional Pathlet-VAE.

C.2. Generation with Noisy Paths

As mentioned earlier, our framework incorporates the concept of pathlet dictionary and the sparse combination of elements within the dictionary for data generation. This design not only provides interpretability but also introduces sparsity, making the model inherently robust to noise and suitable for data denoising tasks. Specifically, after completing the model training, the dictionary D is fixed as one of the model parameters. For a new noisy data $\boldsymbol{x}$, denoising consists of two steps:

(1) Map $\boldsymbol{x}$ to a new representation space using D. To be specific, representation vector $\boldsymbol{r}$ is obtained by solving:

$$\min_{\boldsymbol{r}_i \in \{0,1\}} \quad \lambda_2 ||\boldsymbol{r}||_1 + \lambda_3 ||\boldsymbol{x} - D\boldsymbol{r}||_1 \tag{16}$$

This problem can be viewed as a simplified version of the original problem because the dictionary is fixed at this moment. We solve it using the same strategy described before: first get the optimal fractional solution $\boldsymbol{r}^*$ using gradient descent and then round it to get the final binary solution $\boldsymbol{r}^-$.

(2) Once we get $\boldsymbol{r}^*$, then the denoised data $\hat{x}$ can be obtained by multiplying D and $\boldsymbol{r}^-$.

C.3. Train Algorithm

Here we detail the two-step algorithm outlined in Section 4.4 (Training Procedure). We employ a relax-and-round approach: the binary constraints are relaxed to the continuous domain [0,1] for gradient-based optimization via projected gradient descent, followed by probabilistic rounding to recover the discrete solution.

Algorithm 1 Training Procedure

```
Input: $X$: path matrix; $\epsilon, \theta$: hyperparameters; $L$: loss function; $\delta$: learning rate;
Output: Solution $R^r$ and $D^r$
 1: # Step1, we compute the fractional solution $R^*$ and $D^*$ using gradient descend.
 2: Initial $R_0 = I$; initial $D_0 = X$, initial $\Delta = +\infty$
 3: Randomly initialize the parameters $\alpha, \beta$ of the VAE model.
 4: while $\Delta < \epsilon$ do
 5:     Compute loss: $L_k = L(\alpha_k, \beta_k, R_k, D_k)$ by Eq 15
 6:     Compute gradient and update the parameters
            $R_{k+1} = R_k - \delta \nabla_{R_k} L_k \quad D_{k+1} = D_k - \delta \nabla_{D_k} L_k$
            $\alpha_{k+1} = \alpha_k - \delta \nabla_{\alpha_k} L_k \quad \beta_{k+1} = \beta_k - \delta \nabla_{\beta_k} L_k$
 7:     Clip the result to ensure that every element in $R_k$ and $D_k$ has a value between 0 and 1
 8:     $\Delta = |L_k - L_{k-1}|$
 9: end while
10: #Step2, round solution $R^r$ and $D^r$ based on $R^*, D^*$.
11: Sample $R$ with $P(R_{i,j} = 1) = min(1, \theta R^*_{i,j})$
12: Sample $D$ with $P(D_{i,j} = 1) = min(1, \theta D^*_{i,j})$
```

C.4. Post-processing Algorithm

Our approach essentially involves learning the distribution of a given dataset and then sampling from it to generate new paths. Therefore, an unavoidable issue is that the connectivity of the generated paths is not strictly guaranteed. To address this problem, we propose a greedy algorithm that uses road network information to post-process the generated paths to ensure connectivity.

The detailed steps are explained in Algorithm 2: Given a network G and a generated path on it, assume it is not a completely connected path but a set of separate segments. Our goal is to complete these segments into a full path. The basic idea of the algorithm is: For a set of separate segments, each time find the two segments with the smallest connection cost (The connection cost is defined as the length of the shortest path in the G that connects these two segments), merge them into a new segment, and repeat this process until a complete path is obtained.

Algorithm 2 Post-processing Procedure

```
Require: List of path fragments segments; Network $G$; Cost function cost(path)
Ensure: Merged path merged_path
 1: while Number of fragments in segments $> 1$ do
 2:     Initialize minimum cost min_cost $\leftarrow \infty$
 3:     Initialize best fragment pair best_pair $\leftarrow$ None
 4:     for Each pair of fragments $(s_i, s_j)$ in segments, where $i \neq j$ do
 5:         Compute the minimum cost to connect $s_i$ and $s_j$:
 6:         Use the shortest path algorithm to find the shortest path between the start or end points of $s_i$ and $s_j$
 7:         Obtain the minimum cost cost
 8:         if cost $<$ min_cost then
 9:             Update min_cost $\leftarrow$ cost
10:             Update best_pair $\leftarrow (s_i, s_j)$
11:             Record the shortest path
12:         end if
13:     end for
14:     Merge best_pair using shortest path
15:     Add new_segment to segments
16:     Remove $s_i$ and $s_j$ from segments
17: end while
18: return merged_path
```

D. Detailed Experimental Setup

D.1. Datasets

Shenzhen. [26] released this dataset containing approximately 510k dense trajectories generated by 14k taxi cabs in Shenzhen, China, which can be downloaded at [27].

Porto. This dataset describes trajectories performed by 442 taxis running in the city of Porto, Portugal [28]. Each taxi reports its location every 15s. This dataset is used for the Trajectory Prediction Challenge@ ECML/PKDD 2015.

D.2. Data Preprocessing.

For these two datasets, we remove trajectories with less than 20 GPS sample points and use the method proposed in [29] to convert trajectory to a series of edges on roadmap. Then the matrices D and X are generated as described in Methodology.

Noise Simulation. To validate the robustness of our method, we chose to manually introduce varying levels of noise into the original dataset. Specifically, we added a node named "unknown" to the map, which is connected to all other nodes. For each trajectory x in the dataset, the probability that each node is randomly replaced by the "unknown" node follows a Bernoulli distribution with parameter p_{noise}. By adjusting p_{noise}, we can simulate noisy datasets at different noise levels.

D.3. Evaluation Protocol

To evaluate whether our framework can effectively learn the data distribution, we use the JSD score to quantitatively measure the distribution distance between the generated trajectory dataset and the real dataset in our experimental section. Specifically, we adopt the same definition as in [1]:

$$\text{JSD }(x_0) = \frac{1}{2}\left[\mathbb{E}_{P_d(e \in x_{1:L}|x_0)} \log \frac{P_d(e \in x_{1:L}|x_0)}{A} + \mathbb{E}_{\mathcal{G}(e \in \hat{x}_{1:L}|x_0)} \log \frac{\mathcal{G}(e \in \hat{x}_{1:L}|x_0)}{A}\right]$$

Here, $\mathcal{G}$ represents the real trajectory distribution, and $P_d\left(e \in x_{1:L} \mid x_0\right)$ is the conditioned probability that e is a part of the trajectory given x_0 as origin edge. A is the average of these two distributions. A smaller JSD value indicates better trajectory modeling performance because the JSD quantifies the distance between distribution.

D.4. Baselines

1) GDP is a diffusion-based model for path planning and generation that learns path patterns while incorporating road network constraints, which can be considered the state-of-the-art method in road network trajectory generation [2].

2) MTNet is a deep generative model which employs a knowledge-based meta-learning module to learn generalized distribution patterns from skewed trajectory data [1].

3) Binary VAE. This basic binary version VAE is identical to the VAE component in our framework to ensure a fair comparison. It directly learns the distribution of vectorized trajectory data and generates datasets accordingly.

4) L1B-VAE. This is an improved algorithm based on the original Binary VAE by adding L1 regularization as a penalty term.

5) DiffWave. This is a diffusion model based on the WaveNet architecture [25]. It employs a novel denoising diffusion process that iteratively refines random noise into coherent patterns, effectively modeling complex data distributions. This method is commonly used as a baseline reference in previous studies within the trajectory generation field.

6) DiffTraj. This is an advanced trajectory generation method based on a diffusion probabilistic model [24], designed to effectively simulate and generate high-quality GPS trajectories. Its innovation lies in the adoption of a novel spatial-temporal diffusion process, which gradually transforms random noise into coherent trajectory patterns, thereby achieving precise modeling of complex data distributions.

D.5. Implementation Details

Experimental Configuration. We randomly sampled 30% trajectories as our test dataset, and use the rest 70% as the training dataset. Our method is implemented in Python and trained using an Nvidia A40 GPU. All experiments are run on the Ubuntu 20.04 operating system with an Intel Xeon Gold 6330 CPU.

Hyperparameter Settings. The dictionary size is set to 1000 for both datasets. The regularization parameter λ is determined using the theoretical optimal hyperparameter selection method proposed in [30]. The VAE latent dimension is set to 64. Training is performed for 200 epochs with a batch size of 32 using the Adam optimizer with learning rate 0.001.

E. Additional Experimental Results

E.1. Data Denoising Results

This section demonstrates the results of data denoising using the method described above. Here, the true dataset is denoted as X, and the noise-contaminated dataset is denoted as $\tilde{X}$. The model receives X as input and outputs the denoised result $\hat{X}$. The error ratio is used as the evaluation criterion, which is calculated as the difference between X and $\hat{X}$ divided by the L1 norm of X. From Figure 6, we can see that as the noise increases, the error rate of the restored data is significantly lower than that of the noisy dataset, indicating that our method effectively removes noise. Especially on the Porto dataset, up to about 68% of the noise can be removed.

E.2. Case Study: Conditional Trajectory Prediction

Figure 7 presents a case study on the Shenzhen dataset: the first half of the path lies on one of the city's main east–west thoroughfares. In the southern region, tourist attractions, dining, and entertainment facilities are densely concentrated, leading to two main trajectory possibilities: going straight or turning left.

Figure 7(b) illustrates how behavioral patterns on this road segment vary over time: during the morning rush hour, lunchtime, and dinner, there is a significant increase in the probability of people turning left towards the southern area, while at other times, east–west commuting is predominant. These results show that our

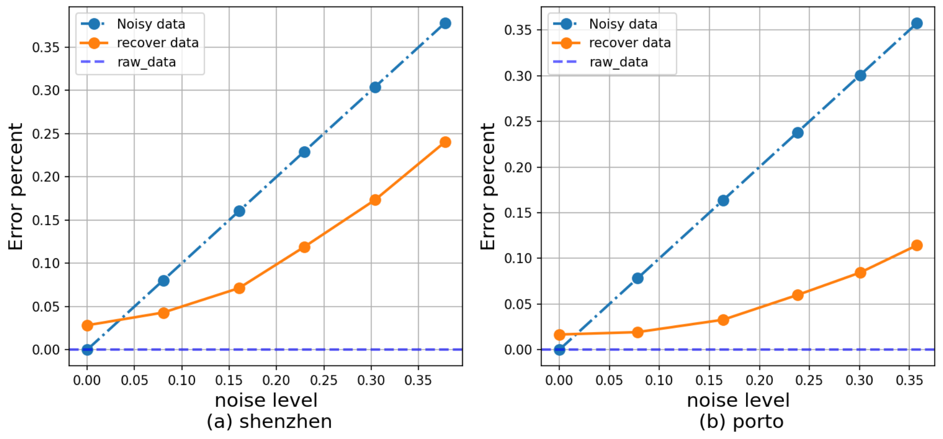

Figure 6: Recovery performance under different noise levels.

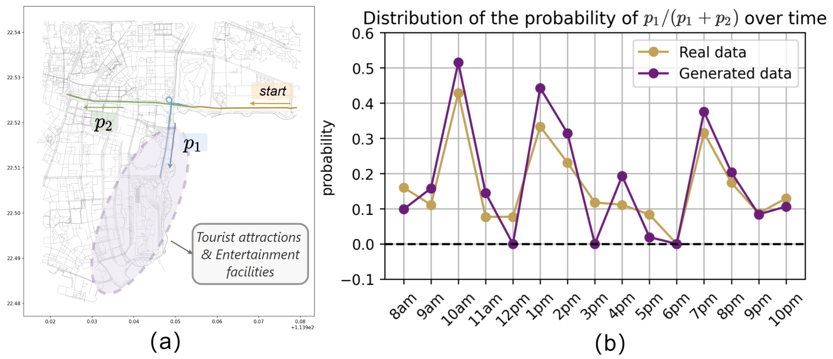

Figure 7: (a) Visualization of conditionally generated trajectories. (b) Probability distribution of path selection over time.

proposed model has effectively learned the underlying spatiotemporal distribution patterns and can capture multi-modal route choices under different temporal conditions.

Table 5: Summary of Mathematical Notation

Basic Variables	
X	Trajectory dataset
Y	Generated trajectory data
$\boldsymbol{t}$	Individual trajectory
$\boldsymbol{x}$	Binary vector representation of trajectory
$\boldsymbol{r}$	Binary representation vector using pathlet dictionary
$\boldsymbol{z}$	Latent Gaussian variable
D	Pathlet dictionary matrix
R	Representation matrix
M	Generative model
Graph & Network	
G	Directed graph (road network)
E	Set of edges in graph
V	Set of vertices in graph
$\|E\|$	Number of spatial units (edges or grid cells)
e	Edge in road network
c	Grid cell
p	Pathlet (trajectory segment)
Model Parameters	
n	Dictionary size (number of pathlets)
K	Dimension of latent variable $\boldsymbol{z}$
α, β	Neural network parameters
λ_1, λ_2	Regularization hyperparameters
θ	Rounding parameter
ϵ	Gaussian noise term
σ^2	Variance parameter
p_{noise}	Noise probability parameter

Matrix & Vector Operations	
$D^{(j)}$	j-th column of matrix D
R_j	j-th row of matrix R
R_{ji}	(j, i)-th element of matrix R
$\boldsymbol{x}_i$	i-th element of vector $\boldsymbol{x}$
$\|\cdot\|_0$	L_0 pseudo-norm (number of non-zero elements)
$\|\cdot\|_1$	L_1 norm
$\|\cdot\|_2$	L_2 norm (Euclidean norm)
Functions & Operators	
$\mathbf{1}_{\{\cdot\}}$	Indicator function
$\max_i$	Maximum over index i
$\sum_{j=1}^{n}$	Summation from $j = 1$ to n
$\exp(\cdot)$	Exponential function
$\log(\cdot)$	Logarithm function
$\text{sigmoid}(\cdot)$	Sigmoid activation function
$f_{\alpha^m}(\cdot)$	Neural network function for parameter m
$f_{\alpha^p}(\cdot)$	Neural network function for parameter p
Loss Functions & Objectives	
L_{VAE}	Variational autoencoder loss
L_{dict}	Dictionary learning loss
$L(H)$	Model description length
$L(D\|H)$	Data description length given model
MDL	Minimum description length
$\|\mathcal{A}_{\text{eff}}\|$	Number of effective dictionary atoms