FACT: High-Fidelity and Controllable Trajectory Data Generation

Yuchen Jiang[1], Xuetao Wei[1], Shiyao Zhang[1], and James Jianqiao Yu[2]
[1]Southern University of Science and Technology, China
[2]University of York, United Kingdom
12232418@mail.sustech.edu.cn, {weixt, zhangsy}@sustech.edu.cn, james.yu@york.ac.uk

***Abstract*—Trajectory data is replete with spatial-temporal information that reflects the traffic conditions within a city, serving as a rich resource for traffic-related tasks. The generation of trajectory data is essential for accurately modeling real-world traffic scenarios. Existing methods for spatial-temporal data generation encounter several challenges, including low fidelity and time-intensive processes. In this paper, we propose a high-Fidelity And Controllable Trajectory generation method (FACT) that employs the diffusion model to achieve high-quality data generation. We design a trajectory denoising transformer architecture, named TDFormer, for high-fidelity trajectory generation. Besides, we define the condition variable to effectively guide the controllable trajectory generation. Furthermore, we propose the adaptive resampling strategy to optimize the efficiency of FACT for practical applications. The resampled trajectory sequence and well-defined condition variables are merged into TDFormer to synthesize geographic trajectories from random noise. Empirical experiments that have been conducted on three distinct real-world datasets provide compelling evidence that FACT is capable of effectively generating high-fidelity and controllable trajectory data by given conditions. Moreover, the generated data can be seamlessly integrated into traffic-related downstream tasks, significantly enhancing the capabilities of database systems for large-scale applications. It also enriches smaller databases, enabling diverse data-intensive applications to address various real-world challenges.**

***Index Terms*—Diffusion transformer, trajectory generation, controllable generation.**

I. Introduction

With the rapid advancement of GPS-embedded devices and data collection technologies, an immense volume of trajectory data has been generated [1]. Defined as a series of spatial points that trace vehicle movement over time, trajectory data is widely applied to various traffic-related tasks due to its rich spatial-temporal insights, such as travel time prediction [2], origin-destination analysis [3], and other location-based services [4]. These applications play a pivotal role in fostering the adaptive evolution of Intelligent Transportation Systems (ITS). However, collecting such extensive datasets poses significant challenges, including privacy concerns, high acquisition costs, and stringent regulatory restrictions on data sharing. Moreover, trajectory data varies across individuals and is highly sensitive to temporal and spatial contexts, complicating efforts to accurately capture the mobility patterns of vehicles. This underscores the pressing need for an efficient trajectory generation method that not only captures the underlying data distributions and features but also enables controllable trajectory generation to facilitate further analysis.

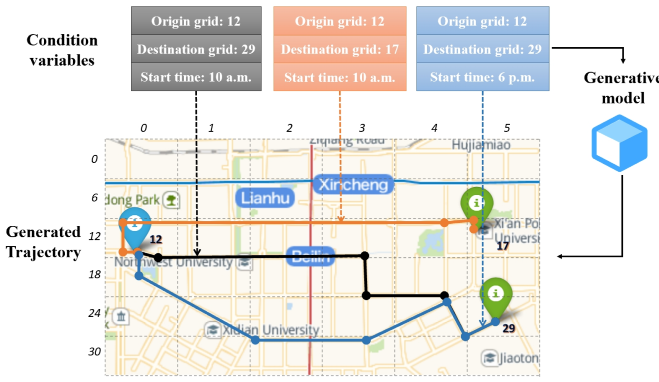


Fig. 1: High-fidelity and controllable trajectory generation.

Several existing data generation methods that utilize deep learning techniques have been developed to produce synthetic data that closely resembles the original, preserving both feature space and data distribution. By maintaining the integrity of the original data distribution, generative methods provide an effective solution for data sharing, offering synthetic results in place of raw data and addressing privacy concerns. Moreover, compared to other types of data, such as text and images, trajectory datasets are inherently sparse and often difficult to collect at a large scale [5]. This scarcity presents significant challenges for downstream applications, which are typically data-intensive and require extensive, high-quality datasets for training. By generating synthetic trajectories, generative models can substantially expand trajectory databases, enhancing their capacity to support applications like mobility prediction and spatiotemporal analysis. These models enable participants—companies, organizations, and individuals—to simulate real-world traffic patterns with high fidelity, meeting the needs of data-hungry trajectory systems without direct access to sensitive raw data. Furthermore, controllable data generation presents a promising approach for large-scale data compression and alleviates storage challenges based on well-defined condition variables.

To generate trajectories that reflect the complexity of real-world scenarios, researchers have explored a variety of approaches. Early efforts, rooted in rule-based or statistical frameworks, offered interpretability but struggled to capture the intricate spatiotemporal dynamics of human mobility [6]. Subsequently, deep learning techniques, such as Generative Adversarial Networks (GANs) [7], Variational Autoencoders (VAEs) [8], and Diffusion Models [9]–[11], have been employed to model complex trajectory distributions. Despite their progress, these methods, commonly observed in trajectory modeling approaches, emphasize general trends in vehicle transitions by simplifying trajectories into coarse representations, such as grids [12]–[14], or converting them into images [15]–[17], leading to coarse-grained generation [18].

Additionally, existing approaches, like Diff-RNTraj [19] and DiffTraj [9], lack robust mechanisms for controllable generation, particularly in defining and applying condition variables to guide trajectory data generation. As a result, challenges remain in generating fine-grained, condition-specific trajectory data. The primary challenges can be summarized as follows:

- **Fidelity**: Current generative methods struggle to consistently produce high-quality trajectory data due to the intricate spatial-temporal features inherent in such datasets. Additionally, road network topology must be incorporated to ensure high-fidelity trajectories suitable for traffic-related analyses. Existing approaches often require pre-training steps to encode the topology information and rectify generated outputs. Furthermore, beyond macro-scale transitions, real-time trajectory modeling demands high-resolution generation for fine-grained analysis, underscoring the need for high-fidelity data generation in trajectory synthesis tasks.
- **Controllability**: Most generative models are trained on comprehensive datasets representing entire cities without targeted guidance, focusing predominantly on group mobility simulation rather than precise individual trajectories at the GPS point level. Generating data tailored for specific traffic-related tasks or regions remains a challenge, resulting in uncontrollable outputs. While some generative methods employ conditional variables to guide training and inference, applying such strategies to trajectory data is difficult due to its implicit and unlabeled nature. Unlike images or other labeled datasets, trajectory data lacks explicit descriptors, making it challenging to guide the generation process with controllable precision.
- **Efficiency**: Achieving a balance between model performance and computational demands is another critical challenge. While diffusion models excel in generating high-quality data compared to GANs and VAEs, their training and inference times are significantly longer, raising efficiency concerns. Similarly, transformer architectures, widely used for handling time-series data, demand substantial computational resources due to the attention mechanism. This necessitates further exploration into optimizing both the efficiency and quality of trajectory generation systems.

To address these limitations, we propose **FACT**, a high-**F**idelity **A**nd **C**on**T**rollable generative model that employs the diffusion model to generate trajectory data as illustrated in Figure 1. FACT introduces a Trajectory-Denoising transFormer architecture, **TDFormer**, tailored for generating high-fidelity trajectory data. Furthermore, condition variables are designed to comprise multiple features to accurately represent trajectory characteristics, effectively guiding the training process and facilitating the generation of controllable outputs while reducing extraneous features for enhanced efficiency. To mitigate the computational costs associated with transformer structures during training and inference, we propose an adaptive resampling strategy. This strategy not only accelerates inference speed but also optimizes overall model efficiency by reducing model complexity through adaptive trajectory data resampling. The resampled trajectory data, paired with condition variables, is then integrated into the TDFormer to synthesize geographic trajectories from random Gaussian noise.

Extensive experiments conducted on three real-world datasets validate the efficacy of FACT. Results demonstrate its ability to generate high-fidelity trajectories suitable for traffic-related analyses while ensuring controllable outputs for targeted applications. The contributions of this paper are summarized as follows:

- We propose the FACT framework based on TDFormer for generating high-fidelity trajectories. The synthesized trajectories preserve the data distribution of the original dataset, effectively reflecting real-world traffic scenarios and supporting diverse downstream tasks.
- We design the condition variable tailored for trajectory data, enabling controllable trajectory generation with efficient representation for trajectory data. Additionally, we introduce an adaptive resampling strategy to optimize the training and generation process, enhancing the model's efficiency and flexibility.
- We validate FACT using three real-world datasets, demonstrating superior performance in generating high-fidelity trajectory data compared to baseline methods. Further experiments on downstream tasks verify the utility of the synthesized trajectories.

The remainder of this paper is structured as follows: Section II reviews related work on generative models, conditional generation, and model efficiency. Sections III and IV present the problem definition and the proposed methodology, respectively. Section V details the experimental setup and compares the performance of FACT with state-of-the-art models. Finally, Section VI concludes the paper and outlines future research directions.

II. Related Work

In this section, previous work about the generative model, conditional generation, and model efficiency are discussed and summarized. We focus on the diffusion model for its notably better performance over other generative models.

A. *Diffusion Model for Trajectory Generation*

As a state-of-the-art data generation technique, the diffusion model showcases its robust capabilities in generating data [20]–[22]. The diffusion model operates through two primary processes: the forward process and the reverse process. In the forward process, noise is gradually added to the original data, while the reverse process learns to reconstruct the original data from the noisy version. Several advancements have been made to enhance both the speed and quality of generation. The Denoising Diffusion Implicit Model (DDIM) [23] accelerates sampling through a non-Markovian diffusion process. Learning the variances in the reverse diffusion process further accelerates the forward process with minimal loss in sample quality [24]. Additionally, numerous studies have explored spatial-temporal generation using the diffusion model. For example, DiffTraj [9] employs the diffusion model with a U-Net architecture to generate trajectory data. ControlTraj [11] extends this approach by incorporating topology constraints to produce higher-quality data. Diff-RNTraj [19] focuses on road segments and introduces a pretraining module to improve their representation. Moreover, the transformer architecture has also demonstrated its superior ability in data generation [25].

B. *Conditional Generation*

Conditional data generation has become a crucial area in machine learning, especially through the use of Conditional Generative Adversarial Networks (CGAN) [26] and Conditional Variational Autoencoders (CVAE) [27]. Both CGAN and CVAE enhance the capabilities of original models by conditioning them on additional information, such as class labels or attributes, enabling the generation of data that satisfies specific conditions. A key advantage of conditional generation is its ability to produce high-quality synthetic data, which can enrich limited datasets. For example, Das et al. propose a hybrid model that combines a conditional generative flow with a classifier to generate synthetic data, helping address the challenges of limited and scarce labeled data during the pandemic [28]. Furthermore, diffusion models can provide guidance for the given conditions [29]. Condition variables are embedded and concatenated with hidden variables during training [9], or they can interact with the model training through mechanisms like cross-attention [25], [30]. This allows the model to be directed by the condition variables, enabling the generation of targeted and desired outcomes.

C. *Model Efficiency*

Although the diffusion model outperforms other methods with higher generation quality, there remains a challenge in balancing computational cost and model performance [31]. Due to the Markov-based forward and reverse processes, both the training and generation steps must be carried out sequentially. Furthermore, to ensure effective training, the number of steps should be sufficiently large to allow the noised data to closely follow a Gaussian distribution. As a result, the computational cost increases with the number of steps. Additionally, specific architectures, such as transformers, demand more computational resources and have higher inference time overhead due to their global attention mechanism [32], [33]. Therefore, optimizations are necessary for both training and inference to make the model more efficient in real-world applications.

III. PRELIMINARIES

In this section, we introduce key preliminaries to define the problem and establish the foundational concepts. Additionally, we provide an overview of the diffusion probabilistic model and the diffusion transformer.

A. *Problem Definition*

Definition 1 (GPS Trajectory) *Let $\mathcal{P}$ denote a sequence of consecutively sampled GPS points, i.e., $\mathcal{P} = \{p_1, \ldots, p_m\}$, where $p_i = [lon_i, lat_i, t_i]$, $i \in \{1, \ldots, m\}$, represents the latitude and longitude of the GPS point at the time step t_i, indicating the user's location at that time.*

Definition 2 (Traffic road network) *The traffic road network, which reflects the topology of the city, is defined as a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. $\mathcal{V}$ contains the intersections between the road segments and $\mathcal{E}$ contains all the road segments. The road topology information can be obtained publicly, e.g., OpenStreetMap*[1].

Definition 3 (Map-matched trajectory) *The raw GPS trajectory dataset collected from mobile devices is randomly offset against real trajectory data. Map-matching algorithms, like fast map-matching [34], are applied to the raw trajectory dataset to get map-matched trajectory data for high-quality traffic-related tasks. Let MM denote the map-matching algorithm and $\hat{\mathcal{P}}$ denote the map-matched trajectory, then $\hat{\mathcal{P}} = MM(\mathcal{P})$, where $\hat{\mathcal{P}} = \{\hat{p}_1, \ldots, \hat{p}_m\}$ and $\hat{p}_i = [\hat{lon}_i, \hat{lat}_i, t_i]$, $i \in \{1, \ldots, m\}$.*

Definition 4 (Trajectory condition variable) *Let $\mathbf{c}$ denote the condition variable for corresponding trajectory data. Here condition variable $\mathbf{c} = \{f_1, f_2, \ldots\}$, where f_i is i-th feature, is a vector consisting of several features about the trajectory and can be easily grasped by the generative model. Besides, the condition variable should efficiently reveal the trajectory conditions without duplicates or similar features.*

Problem 1 (Trajectory Generation) *Given the original set of GPS trajectories $\mathcal{T} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n\}$, where $\mathcal{P}_i = \{p_1^i, \ldots, p_m^i\}$ is a trajectory. The task is to train a generative model $\mathbf{G}$ to generate several trajectories $\mathcal{T}_k = \mathbf{G}(\mathbf{C}_k)$, where $\mathbf{C}_k$ denotes the condition set for generated trajectories and k denotes the number of the generated trajectories, that have similar data distribution to $\mathcal{T}$ with high fidelity according to the corresponding city's road network $\mathcal{G}$. Besides, the generation results should be correctly controlled by given conditions $\mathbf{C}_k$. Furthermore, the computational cost should be acceptable in real-world challenges.*

[1]http://www.openstreetmap.org/

B. Diffusion Probabilistic Model

The diffusion probabilistic model, commonly referred to as the diffusion model, is an effective model for generating high-quality data, including images, text, and audio. It offers remarkable flexibility and data modeling capabilities, allowing it to address a variety of tasks such as image generation, super-resolution, image restoration, editing, as well as natural language processing and multimodal learning. This makes it superior to many other models in terms of performance and versatility [21]–[23]. The core idea of the diffusion model revolves around two main processes: the forward process and the reverse process. In the forward process, random noise is progressively added to the original data, whereas, during the reverse process, the model learns to reconstruct the original data from the noisy version. These processes can be mathematically formulated as follows.

Forward process: Given the original data X_0, Gaussian noises are added on X_0 through T steps. The forward process can be defined as a Markov chain:

$$q(X_{1:T}|X_0) = \prod_{t=1}^{T} q(X_t|X_{t-1}), \tag{1}$$

$$q(X_t|X_{t-1}) = \mathcal{N}(X_t; \sqrt{1-\beta_t}X_{t-1}, \beta_t\mathbf{I}), \tag{2}$$

where $\mathbf{I}$ is identity matrix and $\beta_t \in (0,1)_{t=1}^{T}$ is corresponding variance. Since backward propagation can not be applied directly on samples from Gaussian distribution, reparameterization trick is applied [21], which represents X_t as $X_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0,\mathbf{I})$ and $\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$.

Reverse process: During the reverse process, the model learns to recover the original data from noised data. The noised data is defined as $X_t \sim \mathcal{N}(0,\mathbf{I})$. The reverse process can also be defined as a Markov chain:

$$p_\theta(X_{0:T}) = p(X_T)\prod_{t=1}^{T} p_\theta(X_{t-1}|X_t), \tag{3}$$

$$p_\theta(X_{t-1}|X_t) = \mathcal{N}(X_{t-1}; \mu_\theta(X_t,t), \sigma_\theta(X_t,t)^2\mathbf{I}), \tag{4}$$

where $\mu_\theta(X_t,t)$ and $\sigma_\theta(X_t,t)$ are mean and variance by θ respectively. Introduced in [21], for any $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t(t>1)$ and $\tilde{\beta}_1 = \beta_1$, μ_θ and σ_θ are parameterized as:

$$\mu_\theta(X_t,t) = \frac{1}{\sqrt{\alpha_t}}(X_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(X_t,t)), \tag{5}$$

$$\sigma_\theta(X_t,t) = \sqrt{\tilde{\beta}_t}. \tag{6}$$

In practice, the training process of the diffusion model can be summarized as learning the Gaussian noise ϵ_θ and minimizing the mean squared error (MSE) between the predicted noise $\epsilon_\theta(X_t)$ and the true noise ϵ_t, which is sampled from a Gaussian distribution. This objective can be expressed as follows:

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \mathbb{E}_{t,X_0\sim q}\left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta\left(X_t,\ t\right)\right\|_2^2, \tag{7}$$

where $\boldsymbol{\epsilon}_t$ is the true noise for time step t.

C. Diffusion Transformer

The Diffusion Transformer (DiT) [25] combines diffusion probabilistic models with Vision Transformer (ViT) [35] backbones to tackle high-fidelity generative tasks, especially excelling in image synthesis. Its architecture uses a sequence-based transformer design, treating data as embedded tokens, and incorporates temporal conditioning through trainable embeddings that encode information specific to the diffusion step. Notable improvements include cross-attention mechanisms for conditional generation tasks and scalable transformer layers capable of adapting to different input resolutions and dimensions.

The generative process in DiT follows the diffusion framework, with a forward Markovian noise addition and reverse denoising achieved via parameterized transformer networks. An encoder E, derived from a pretrained AutoEncoder, first encodes a given image before passing it into the diffusion model for further training. To handle the computational intensity, the encoder compresses images into a latent space, reducing the input data's dimension while preserving its representational capacity. The encoded images are then split into patches and processed as sequences for the ViT structure. The diffusion model is trained on $z = E(X)$, with X denotes input data and E is frozen during training. The images are generated by sampling z' from the diffusion model and then decoded using the pretrained decoder D. The final synthesized image is $X' = D(z')$. DiT demonstrates superior performance in high-resolution image generation, setting a new standard for generative models in complex data tasks.

IV. High-Fidelity and Controllable Trajectory Data Generation Framework

In this section, we introduce the proposed model FACT for generating high-fidelity and controllable trajectories. We propose an efficient framework for traffic trajectory generation. The following details outline the design of FACT.

A. TDFormer Block

As discussed in Section III, the goal is to design a neural network capable of accurately predicting the noise $\epsilon_\theta(X_t,t)$ at each time step t. While the transformer architecture has demonstrated superior performance [25], it requires adaptation to handle the trajectory-specific conditions of trajectory data. To address this challenge, we propose the TDFormer block, a novel architectural design that integrates adaptive Trajectory Layer Normalization (adaTLN) with Multi-Head Self-Attention (MHSA), enabling precise modeling of trajectory conditions and spatial-temporal features. The adaTLN mechanism is specifically tailored to provide conditional guidance, dynamically adjusting normalization based on trajectory-specific contexts, while MHSA captures complex dependencies within the trajectory sequences. This synergistic design ensures a robust and accurate representation of trajectory information. Specifically, the TDFormer block includes two primary modules, Multi-Head Self-Attention (MHSA) and an adaTLN tailored for conditional guidance. MHSA allows

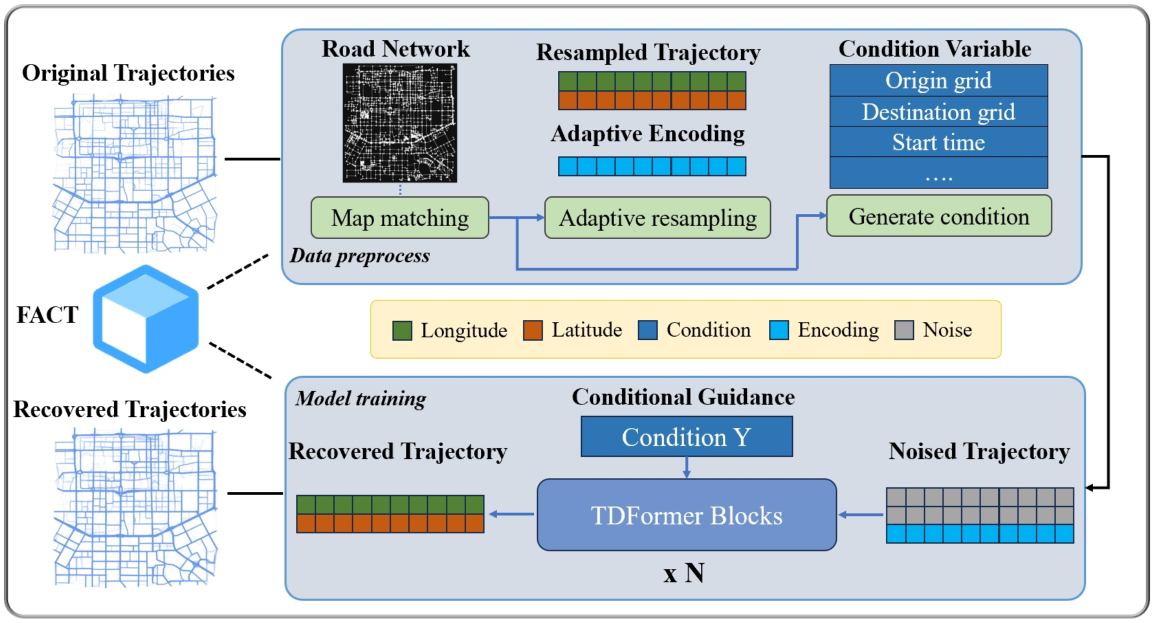


Fig. 2: Network structure of FACT.

for the fusion of multi-scale features through the multi-head settings and efficiently captures both short-term and long-term dependencies within trajectory data. The adaTLN mechanism utilizes adaptive normalization layers [25], introducing conditional information about the trajectory, thereby improving the quality and controllability of the generated data. In practice, the adaTLN mechanism is implemented via a scale and shift module. Embeddings from the time step t and the condition c are used to predict appropriate scale and shift parameters, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, enabling effective condition guidance for generation. We also apply dimension-wise scaling parameters $\boldsymbol{\alpha}$ [25] just before the residual connections within the TDFormer block, scaling the output signal of the module and dynamically adjusting its contribution to the overall output. Additionally, as observed by Goyal et al., zero-initializing the final batch norm scale factor $\boldsymbol{\gamma}$ in each block accelerates large-scale training in supervised learning settings [36]. Therefore, adaTLN initializes the scale and shift parameter as zero to enhance training efficiency. Initializing $\boldsymbol{\alpha}$ to zero during the early stages of training helps suppress the premature influence of new modules, ensuring training stability. As training progresses, $\boldsymbol{\alpha}$ is optimized to adapt to task-specific requirements, allowing the effects of conditional adjustments or other modules to emerge gradually. This scaling parameter enhances the controllability of conditional inputs while improving model training stability and convergence. The flow of these parameters is shown below:

$$\boldsymbol{\alpha}_{1/2}, \boldsymbol{\gamma}_{1/2}, \boldsymbol{\beta}_{1/2} = \mathrm{MLP}(\mathrm{Emb}(\boldsymbol{c}) + \mathrm{Emb}(t)), \tag{8}$$

$$\boldsymbol{h}^{i+1} = \boldsymbol{\alpha}_1 \cdot \mathrm{MHSA}(\text{adaTLN-Zero}(\boldsymbol{H}^i)) + \boldsymbol{H}^i, \tag{9}$$

$$\boldsymbol{H}^{i+1} = \boldsymbol{\alpha}_2 \cdot \mathrm{FFN}(\text{adaTLN-Zero}(\boldsymbol{h}^{i+1})) + \boldsymbol{h}^{i+1}, \tag{10}$$

where $\boldsymbol{\alpha}_{1/2}$ is dimension-wise scale factor, $\boldsymbol{\gamma}_{1/2}$ and $\boldsymbol{\beta}_{1/2}$ are scale and shift factors respectively. MLP and Emb are multi-layer perceptron and embedding layer respectively. FFN denotes Feed Forwad Network. This ensures that the condition variables are properly aligned with the trajectory sequence, resulting in generated trajectories that are both more realistic and context-specific. As illustrated in Figure 3, the outcome of the TDFormer block is computed as $\boldsymbol{H}^{i+1} = \mathrm{TDFormer}(\boldsymbol{H}^i)$, where $\boldsymbol{H}^i$ represents the hidden states from the previous step, and $\boldsymbol{H}^{i+1}$ denotes the updated hidden states after applying the TDFormer. This iterative process enables the model to refine and improve trajectory predictions at each step, taking into account both spatial and temporal dependencies, as well as conditional guidance.

B. Road Network Information

To improve the quality of trajectory generation, the focus is to reduce unrealistic and meaningless trajectories, particularly those that are not confined to the road network. GPS offsets during data collection often introduce biases between raw trajectory data and real trajectory data, especially for traffic trajectories that are primarily distributed on road segments, such as bus or taxi trajectories. Rather than incorporating road network topology or pretraining a road embedding network, we concentrate on accurately recovering real trajectory data from Gaussian noise, which ensures that the generated trajectories are not only realistic but also adhere to the spatial constraints of actual road networks thanks to the effective

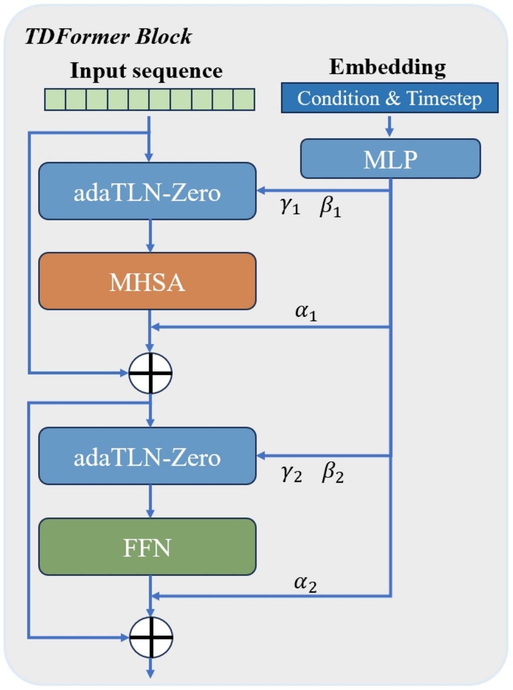


Fig. 3: TDFormer block.

reverse process with TDFormer, leading to more meaningful and contextually relevant results.

Given raw trajectory dataset $\mathcal{T}$, fast map matching [34], is applied to the raw trajectories to get road-constrained dataset $\hat{\mathcal{T}}$ based on the road network $\mathcal{G}$ in the city for subsequent training and analysis. Therefore, considering the map-matched trajectory data, the loss function can be defined as:

$$\mathcal{L}(\theta)_{real} = \mathbb{E}_{t,X_0 \sim q} \left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta \left(\hat{X}_t,\ t \right) \right\|_2^2, \tag{11}$$

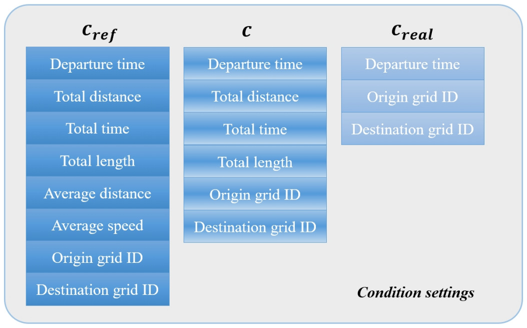


Fig. 4: Different condition settings.

C. Trajectory Condition Design

Carefully designed guidance variables, such as spatial-temporal constraints or semantic tags, are indispensable for effective training pipelines to achieve fine-grained and controllable trajectory generation [11], [27]. These variables provide structured supervision, enabling models to align generated trajectories with user-specified conditions while maintaining high fidelity. Guidance variables are essential for training to achieve controllable generation. Unlike image generation, trajectory data lacks explicit labels that can directly guide the generation process, which typically leads to the use of an unconditional generative model. Therefore, it is crucial to design appropriate condition variables for trajectory data that can effectively guide both the training and inference processes of the generative model. These condition variables help steer the model toward generating realistic and contextually relevant trajectories, allowing for more controlled and meaningful outputs during the generation process. Building upon methods like DiffTraj [9] and ControlTraj [11], our approach refines trajectory condition variables by reducing redundancy and optimizing efficiency. By minimizing unnecessary condition information, our model achieves comparable or superior condition guidance performance while utilizing less input data, thereby enhancing both scalability and applicability. The original condition variable consists of *departure time* (*dt*), *total distance* (*td*), *total time* (*tt*), *total length* (*tl*), *average distance* (*ad*), *average speed* (*as*), *origin grid ID* (*oID*) and *destination grid ID* (*dID*). Thus, we define the condition variable for trajectory data as:

$$\boldsymbol{c}_{ref} = \{dt, td, tt, tl, ad, as, oID, dID\}, \tag{12}$$

where $\boldsymbol{c}_{ref}$ denotes the reference condition variable. Among these features, *departure time*, *origin grid ID*, and *destination grid ID* are discrete features that roughly describe the temporal characteristics of trajectory and the location of the origin and destination points. Other continuous features introduce the travel distance and travel time in detail. These features are appropriate for the condition definition of the trajectory data and are easily adopted by the model.

Based on these features, we extend them into a simplified but efficient version. We retain the discrete features that define the spatial and temporal overview of the trajectory. Besides, total distance, total time, and total length which guide the detailed generation are also retained for training and inference. The remaining features can either be obtained by linear transformation from the selected features or have little effect on the generation results, which is verified in Section V-C. Thus, we can get the updated condition variable as:

$$\boldsymbol{c} = \{dt, td, tt, tl, oID, dID\}. \tag{13}$$

However, in real-world scenarios, we often don't have access to the detailed length or distance of the desired trajectory data beforehand, making these features unknown before generation. Therefore, we define a condition variable that only includes the departure time, origin grid ID, and destination grid ID, which are typically available in practical applications. The simplified condition variable for real-world applications is described as follows:

$$\boldsymbol{c}_{real} = \{dt, oID, dID\}, \tag{14}$$

where $\boldsymbol{c}_{real}$ means condition variable for real-world applications. Further discussion on guiding performance of these condition variable designs are shown in Section V-C.

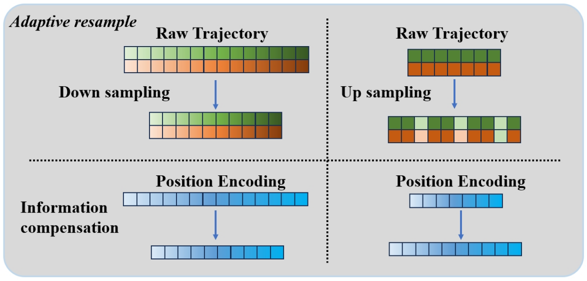


Fig. 5: Adaptive resampling.

D. Adaptive Resampling Strategy

1) Adaptive Resampling: Reducing the model size is crucial for minimizing both time and storage costs. To achieve this, we propose an Adaptive Resampling (AR) strategy for the input data. Given the complexity of the transformer structure, which is influenced by the maximum length of input sequences, we resample the trajectory to a fixed length $\boldsymbol{L}$ for trajectories of varying lengths. This resampling is achieved through linear interpolation. The resampled trajectory can be represented as:

$$\hat{\mathcal{P}}' = \mathrm{RA}(\hat{\mathcal{P}}, \boldsymbol{L}), \tag{15}$$

where $\hat{\mathcal{P}}$, RA, and $\boldsymbol{L}$ denote the map-matched trajectory, the resampling algorithm (linear interpolation by default), and the target length of resampling, respectively. Here, we assume that the trajectory length $\boldsymbol{L}$ remains consistent within a specific city but may vary across different cities. To account for the varying trajectory length distributions of each city, we set $\boldsymbol{L}$ to the average trajectory length of the respective city. This approach ensures the model adapts to the unique characteristics of each city's trajectory data. As illustrated in Figure 5, we employ linear downsampling or linear interpolation to standardize trajectories of different lengths, depending on whether they exceed or fall short of $\boldsymbol{L}$.

2) Information Compensation: However, while resampling offers benefits, it also introduces the risk of information loss, especially for long-distance trajectories where many location points may be omitted. Although the original trajectory length is provided to guide the generation model, it is insufficient for achieving optimal performance. To address this, we propose an Information Compensation (IC) mechanism. We reconstruct the positional encoding within the input sequence to ensure the model receives accurate positional information. Typically, the time difference between two trajectory points p_i and p_j will change when up-sampling or down-sampling, making the original positional encoding less effective in capturing the true relative temporal differences. To resolve this, we design a new positional encoding specifically tailored for resampled sequences. The updated positional encoding can be described as follows:

$$pos_L = \frac{pos}{l_0/L}, \tag{16}$$

$$PE_{(pos_L,2i)} = \sin\left(\frac{pos_L}{10000^{2i/d_{\text{model}}}}\right), \tag{17}$$

$$PE_{(pos_L,2i+1)} = \cos\left(\frac{pos_L}{10000^{2i/d_{\text{model}}}}\right). \tag{18}$$

In these equations, l_0 represents the original trajectory length, pos indicates the original position of each element in the sequence, i denotes the dimension index, and d_{model} specifies the model's dimensionality. The term pos_L corresponds to the new position list adjusted for the fixed length $\boldsymbol{L}$. The scaling factor l_0/L in this context compensates for the degree of resampling applied, ensuring that the positional encoding accurately reflects the true time intervals between points in the resampled trajectory. This adjustment maintains the integrity of the temporal information despite changes in sequence length, thereby improving the model's capacity to effectively process resampled data. The information compensation is illustrated in Figure 5.

E. Trajectory Generation

Based on TDFormer, well-defined condition variables, and the adaptive resampling strategy, FACT is outlined with the following training and generating processes:

1) Training: Given the map-matched GPS trajectory set $\hat{\mathcal{D}}i$, we first resample it into a set $\hat{\mathcal{D}}i'$ with a fixed length $\boldsymbol{L}$ to standardize the input sequence length. Consequently, the input for TDFormer can be formulated as follows:

$$X_0 = \hat{\mathcal{P}}' + PE_L, \tag{19}$$

where $\mathcal{P}$ represents the GPS points and PE_L is the revised positional encoding under the fixed length $\boldsymbol{L}$. The model takes this as input, along with condition embeddings $\boldsymbol{Y}$ and time step embeddings $\boldsymbol{T}$, which are derived from the condition variable $\boldsymbol{c}$ and the time step t, respectively. These condition attributes, as defined in Equation 12, 13, and 14, guide the model in learning the desired trajectory generation patterns, ensuring that the generated trajectories adhere to the provided spatial-temporal constraints and allow for controllable generation. The goal of training is to predict the noise at a given time step t, condition $\boldsymbol{c}$, and resampled length $\boldsymbol{L}$. The optimization function can be represented as:

$$\min_{\theta} \mathcal{L}(\theta) = \min_{\theta} \mathbb{E}_{\boldsymbol{c},t,X_0\sim q} \left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta\left(X_t,\ t,\ \boldsymbol{c}\right)\right\|_2^2, \tag{20}$$

2) Generation: During sampling, we begin with an initial sample $X_T \sim \mathcal{N}(0, \mathbf{I})$, where X_T represents the noisy trajectory at the final time step T, drawn from a standard Gaussian distribution. Then, we recursively sample the trajectory at each previous time step using the model's learned reverse process: $X_{t-1} \sim p_\theta(X_{t-1}|X_t)$, where $p_\theta(X_{t-1}|X_t)$ is the learned conditional distribution that models the transition from X_t to X_{t-1}, given the current noisy data. This process is typically performed using the reparameterization trick, which ensures that the model can generate samples by backpropagating through the noise addition process.

By iteratively applying this reverse process, the model refines the generated trajectory, reducing noise step by step. The result is a high-quality, realistic traffic trajectory that is consistent with both the spatial constraints of the road network and the temporal dynamics of the trajectory data, ensuring that the generated trajectories follow realistic paths and time intervals.

V. Experiments

In this section, we first describe the experimental settings, including datasets, baselines, evaluation metrics, and hyperparameters. Then, we conduct comprehensive experiments on real-world datasets to evaluate the performance of the proposed method, FACT, and address the following research questions:

- **RQ1**: Does the trajectory data generated by FACT demonstrate superior fidelity while maintaining the data distribution compared to state-of-the-art methods?
- **RQ2**: Can FACT be effectively controlled by the given conditions? What is the accuracy of the guidance in generating desired trajectories?
- **RQ3**: How does each module in FACT contribute to the overall generation performance?
- **RQ4**: How does the efficiency of FACT compare to other methods in terms of time and resource consumption?
- **RQ5**: How does FACT benefit real-world database applications?

A. Experimental Settings

We conduct the experiments using PyTorch. The model is trained on four NVIDIA A100 40GB GPUs.

1) Datasets: We evaluate FACT against various baselines on three real-world datasets, which consist of daily taxi trajectories collected over a month in the cities of Chengdu, Xi'an, and Porto.

TABLE I: Statistics of the Real-world Trajectory Datasets.

Dataset	Chengdu	Xi'an	Porto
Trajectory Number	3 731 344	2 255 474	1 414 164
Average Time	13.51 min	16.11 min	12.19 min
Average Distance	3.56 km	3.49 km	3.96 km
Sampling Interval	3 seconds	3 seconds	15 seconds

2) Baselines: We compare the proposed FACT with state-of-the-art generative methods, including Conditional VAE (CVAE) [37], Conditional GAN (CGAN) [26], and diffusion models, DiffWave [38], DiffTraj [9], Diff-RNTraj [19] and ControlTraj [11].

- **CVAE** [37]: A VAE with four convolutional layers and two linear layers is built for trajectory generation. Condition variables are incorporated during training. The model encodes and decodes the trajectories, and the trained decoder is used for generating new samples.
- **CGAN** [26]: A GAN with four convolutional layers and two linear layers is used for trajectory generation. Condition variables are included in the training process. The generator receives random Gaussian noise and attempts to generate fake samples, while the discriminator distinguishes between real and fake samples. The trained generator is used for data generation.
- **DiffWave** [38]: DiffWave employs a Wavenet structure designed for sequence synthesis using dilated convolutions. It uses 16 residual connected blocks, each containing a bi-directional dilated convolution, which is summed using sigmoid and tanh activations before being processed by a 1D convolutional neural network (CNN).
- **DiffTraj** [9]: A diffusion model utilizing convolutional layers and a U-net structure is designed to generate high-quality traffic trajectories. Condition variables are integrated into the training and inference process to guide the generation of trajectories.
- **Diff-RNTraj** [19]: A pre-trained Road-constraiNtTraj (RNTraj) Vectorization module is employed to transform GPS points into road-based representations, converting discrete representations into vectors for training. This ensures that the generated trajectories are constrained to the road network, enhancing fidelity.
- **ControlTraj** [11]: A pretrained Masked AutoEncoder (MAE) encodes the topology information of road segments traversed by trajectory data to improve fidelity. The topology information is used during both training and inference to guide the generation process.

TABLE II: Details of FACT models. We follow DiT [25] model configurations for the Small (S), Medium (M), and Large (L) variants.

Model	Layers N	Hidden size d	Heads
FACT-S	10	64	8
FACT-M	18	128	16
FACT-L	28	128	16

TABLE III: Hyperparameters for FACT.

Parameter	Setting value	Refer range
Diffusion Steps	500	$300 \sim 500$
Skip steps	5	$1 \sim 10$
Hidden dimension	256	≥ 64
β (linear schedule)	$0.0001 \sim 0.05$	–
Batch size	512	≥ 64
Transformer block	18	$10 \sim 28$

3) Evaluation Metrics: We follow the methodology from previous work [11], [39] and use three evaluation metrics to assess the quality of generated trajectories across different models: **Density error**, **Length error**, and **Pattern score**. These metrics are essential for evaluating trajectory generation performance. Specifically, the Jensen-Shannon divergence (JSD) is used to compare the distribution differences between

TABLE IV: Performance comparison of different generative models.

Dataset	Metrics	CVAE	CGAN	DiffWave	DiffTraj	Diff-RNTraj	ControlTraj	FACT
Chengdu	Density (↓)	0.0583	0.0442	0.0136	0.0051	0.0371	<u>0.0031</u>	**0.0006**
	Length (↓)	0.1630	0.1566	0.0311	0.0144	0.1078	<u>0.0097</u>	**0.0044**
	Pattern (↑)	0.5001	0.5219	0.7590	0.8519	0.6094	<u>0.8547</u>	**0.8869**
	Fidelity (↑)	0.931	0.865	0.547	0.430	**0.0**	0.201	<u>0.173</u>
Xi'an	Density (↓)	0.0569	0.0516	0.0208	0.0106	0.0093	<u>0.0070</u>	**0.0014**
	Length (↓)	0.0607	0.0582	0.0313	0.0152	0.0191	<u>0.0134</u>	**0.0069**
	Pattern (↑)	0.6790	0.7815	0.5920	0.7678	0.7940	<u>0.8330</u>	**0.8717**
	Fidelity (↑)	0.844	0.809	0.417	0.356	**0.0**	0.130	<u>0.107</u>
Porto	Density (↓)	0.0525	0.0435	0.0096	0.0072	0.0052	<u>0.0049</u>	**0.0023**
	Length (↓)	0.0560	0.0479	0.0243	0.0215	0.0156	<u>0.0144</u>	**0.0093**
	Pattern (↑)	0.5194	0.6774	0.8150	0.7940	0.8220	<u>0.8319</u>	**0.8650**
	Fidelity (↑)	1.538	1.447	0.601	0.522	**0.0**	0.285	<u>0.196</u>

Bold shows the best performance, and <u>underline</u> shows the second-best. ↓: lower is better, ↑: higher is better. The 0.0 fidelity errors by Diff-RNTraj are contributed by its map-matching post-processing, which is not integrated by others.

the original and generated datasets. A lower JSD indicates higher similarity and, thus, better generation quality. For evaluation, we apply grid-based statistics by dividing the city into 16x16 grids for distribution counting. Additionally, to assess the fidelity of the generation results, we introduce a metric called **Fidelity error**, which compares the distance error between the generated trajectories and map-matched trajectories, verifying the effectiveness of road constraints across all models. We randomly sample 9,000 trajectories from each model and calculate their metrics.

- **Density error**: The density error evaluates the geographic distribution between the original dataset $\mathcal{D}$ and the generated dataset $\mathcal{D}'$. The evaluation is conducted based on discrete grids, which are applied to the city.
- **Length error**: The length error measures the distribution of trajectory distances. It is calculated by computing the distribution difference in geo-distances between consecutive points in the original and generated trajectories.
- **Pattern score**: The pattern score identifies the top-n grids that occur most frequently. A higher pattern score indicates that more of the top-n grids in the generated dataset align with those in the original dataset, signifying higher distribution similarity. Here, n is set to 25.
- **Fidelity error**: The fidelity error measures the distance difference between the generated trajectory data and the map-matched generation results, which use a post-processing technique for ground truth. Geo-distances between raw points and map-matched points serve as the criterion for this calculation.

4) Hyperparameter: In FACT, we employ a sequence of N TDFormer blocks and each operates with a hidden dimension size d. Following the approach in DiT [25], we use standard transformer configurations that jointly scale N, d, and the number of attention heads. Specifically, we use four configurations: FACT-S, FACT-M, and FACT-L. These configurations cover a broad range of model sizes, enabling us to evaluate the scaling performance. The details of these configurations are provided in Table II. We also list the hyperparameters of the model in Table III. Additionally, the reference range for these hyperparameters is based on experimental observations. It is worth noting that, for comparisons with other baselines, we adopt the FACT-M configuration since it achieves the best generation quality while maintaining a relatively modest number of model layers. In contrast, increasing the number of layers beyond the FACT-M configuration yields tiny improvements in generation quality.

We trained the model on three taxi trajectory datasets from Chengdu, Xi'an cities[2] and Porto[3]. The three datasets represent real-world urban trajectories from Chengdu, Xi'an, and Porto, each with distinct characteristics. Chengdu has the largest trajectory count (3,731,344) with relatively short average distances (3.56 km) and durations (13.51 min), representing high-frequency, short trips. Xi'an includes 2,255,474 trajectories with the highest average duration (16.11 min), reflecting more complex travel patterns. Porto, with the fewest trajectories (1,414,164), shows the longest average distance (3.96 km) but the shortest average duration (12.19 min), likely due to its urban layout. These datasets offer diverse scenarios to evaluate the adaptability of trajectory generation models. We exclude short trajectories as they lack sufficient context to represent complete travel patterns. Specifically, trajectories with lengths under 120 in Chengdu and Xi'an, and under 24 in Porto, which correspond to approximately 6 minutes of real-world travel, are removed. To ensure consistency and quality in the data, the remaining trajectories are then processed using adaptive resampling.

B. Overall Performance (RQ1)

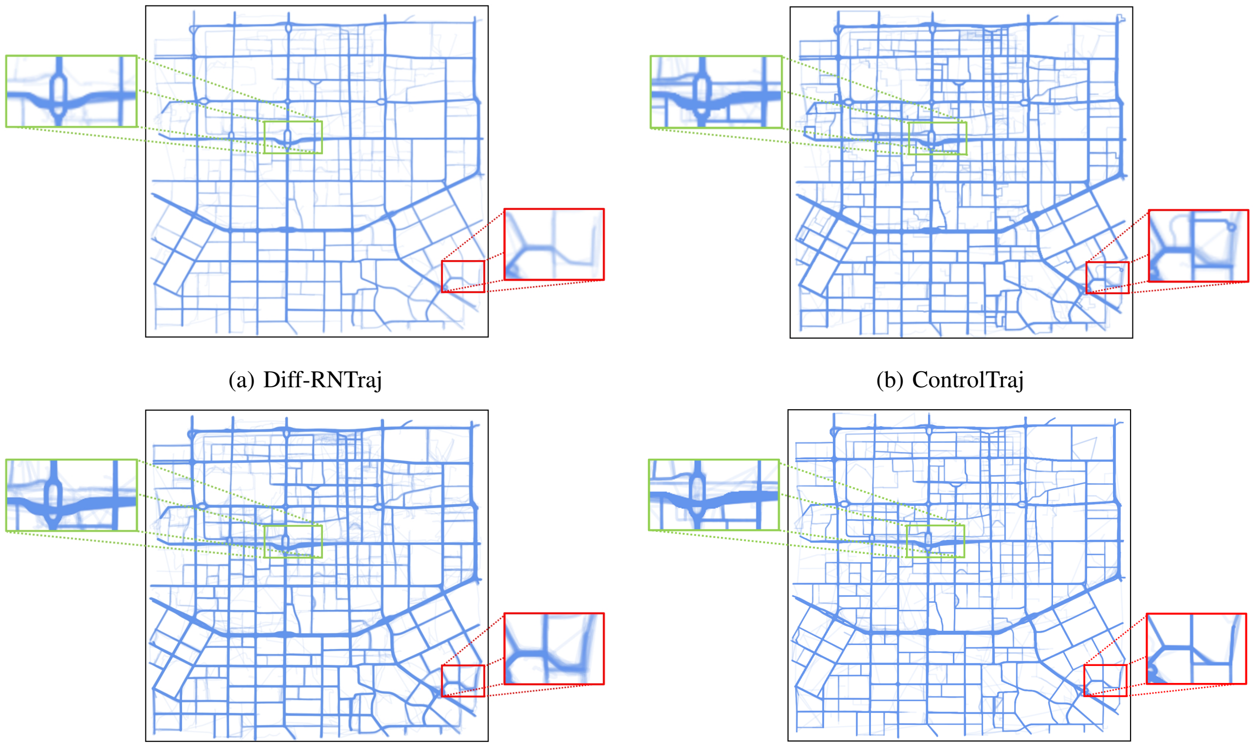
Table IV presents the simulation results of FACT and baselines on the three trajectory datasets. From the results, it can be observed that FACT outperforms other generative models across all datasets by achieving significant improvements in density and pattern metrics. For the Chengdu dataset, FACT reduces the density error and length error by 80.65% and

[2]https://outreach.didichuxing.com/

[3]https://www.kaggle.com/datasets/crailtap/taxi-trajectory/

TABLE V: Controllable generation study.↓: lower is better, ↑: higher is better.

Methods	**CVAE**	**CGAN**	**DiffTraj**	**ControlTraj**	**FACT**
Origin grid accuracy (↑)	55.30%	55.76%	91.59%	93.22%	**94.24%**
Destination grid accuracy(↑)	55.62%	58.03%	91.41%	93.62%	**94.69%**
Total distance error (↓)	1227 m	1203 m	130.1 m	86.10 m	**82.60 m**
Average speed error (↓)	3.480 m/s	3.368 m/s	0.2179 m/s	0.1590 m/s	**0.1384 m/s**


(a) Diff-RNTraj

(b) ControlTraj

(c) FACT

(d) Real

Fig. 6: Visualization of generated trajectory dataset in Xi'an City.

54.64% respectively compared to the best baseline. Similarly, the pattern score improves by 22.16% compared to the best baseline. In the Xi'an dataset, FACT achieves a density error reduction of 80.00% compared to the best baseline and a length error reduction of 48.51% compared to the best baseline. The pattern score increases by 23.17% over the best baseline. For the Porto dataset, FACT demonstrates similar dominance by reducing the density error by 53.06% and reducing the length error by 35.42% compared to the best baseline. The pattern metric improves by 19.69% over the best baseline. These results highlight the effectiveness of the TDFormer architecture compared to the U-Net structure. For both Density error and Length error, FACT achieves the lowest errors in all datasets, demonstrating that the generated trajectories closely match the true distributions in terms of spatial coverage and trajectory length. Moreover, FACT achieves the highest Pattern scores, showcasing its ability to capture the underlying trajectory patterns more effectively than other models. Generative methods like CVAE and CGAN perform worse than diffusion-based methods. Although Diff-RNTraj provides road-constrained trajectory data, it struggles with training and generating, particularly as the number of road segments increases. ControlTraj demonstrates competitive performance but is constrained by its reliance on the U-Net structure. Unlike the TDFormer architecture used in FACT, U-Net is primarily designed for image segmentation tasks and struggles to model complex spatial-temporal relationships effectively. FACT excels in capturing both short-term and long-range dependencies through its MHSA and adaTLN. This limitation reduces its ability to generate high-fidelity and conditionally precise trajectories.

For fidelity, although Diff-RNTraj map-matches its generation results to the road network to produce perfect error scores, its poor performance on other metrics notably undermines this merit. Diff-RNTraj relies on a road-based representation for training and generation but depends on similarity comparisons to recover the GPS sequence, which becomes challenging in complex road network scenarios. As shown in Table IV, Diff-RNTraj performs worse on the Chengdu dataset due to the city's higher number of road segments and less distinguishable road segment differences, resulting in good fidelity but overall poor performance. FACT, which benefited from the superior TDFormer, achieved the second-best performance among all the methods. We also provide the visualization of the generated trajectories in Xi'an City. As shown in Figure 6, three diffusion-based methods are selected to compare the visualization of generation. Diffusion-based methods can generate trajectories with higher similarity and perform well on the whole city and areas of high-density trajectories covered in green boxes. Compared to Diff-RNTraj and ControlTraj, FACT reduces unrealistic and meaningless trajectories and shows higher closeness to the road network, especially on sparse roads covered in red boxes, further improving the generation quality. Although Diff-RNTraj generates road-constrained trajectories, their distribution is different from the original trajectories, leading to poor overall performance. Furthermore, the visualization results illustrate the ability to understand spatial-temporal dynamics for FACT, leading to a more realistic generation.

C. Controllable Generation (RQ2)

Among these generative methods, we select **CVAE**, **CGAN**, **DiffTraj**, and **ControlTraj**, which integrate condition variables to guide both training and generation, to assess the controllability of the generated results. Specifically, we focus on the accuracy of guidance for these methods by sampling 9,000 trajectories in Xi'an City. To evaluate how well the generated trajectories align with the specified attributes, we focus on the origin grid, destination grid, total distance, and average speed to determine if the generated trajectories meet these conditions. For evaluation, we use grid accuracy to assess whether the generated trajectory points are correctly distributed within the target grids for the origin grid and destination grid. To measure total distance and average speed, we apply Mean Absolute Error (MAE). The evaluation metrics are defined as follows: Origin grid accuracy, Destination grid accuracy, Total distance error, and Average speed error. It is important to note that the origin grid and destination grid are explicitly provided in the condition variable $\boldsymbol{c}$, serving as the foundational requirements for condition guidance. Furthermore, since we treat total distance and average speed as redundant features and exclude them from the reference condition variables $\boldsymbol{c}_{ref}$, they serve as hidden comprehension of the condition variables for these methods.

The results in Table V further validate the superior controllability of FACT in traffic trajectory generation. FACT achieves the highest accuracies for both the origin grid accuracy and the destination grid accuracy and the lowest errors for both total distance error and average speed error, surpassing all other models. This indicates that FACT can effectively adhere to conditional constraints, generating trajectories that start and end at the desired locations with remarkable precision. The results also show the reason for the superiority of our model in terms of generation quality. In contrast, traditional models like CVAE and CGAN perform significantly worse, demonstrating their limitations in understanding condition guidance. Notably, while DiffTraj and ControlTraj exhibit competitive performance in certain metrics, they fall short of FACT's overall balance. TDFormer with adaTLN shows its superior performance in aligning generation with condition guidance. This underscores FACT's robustness and its advantage in generating high-fidelity, controllable trajectories without compromising realism.

TABLE VI: Ablation study on FACT.

Metrics	Xi'an		
	Density (↓)	Pattern (↓)	# Params (million)
FACT w/o AR	<u>0.0013</u>	0.8706	8.0518
FACT w/o $\boldsymbol{c}$	0.0049	0.8570	5.4780
FACT-$\boldsymbol{c}_{ref}$	0.0013	**0.8717**	5.7990
FACT-$\boldsymbol{c}$	0.0014	**0.8717**	5.7988
FACT-$\boldsymbol{c}_{real}$	0.0037	0.8689	<u>5.7983</u>
FACT-S	0.0100	0.8378	**1.1231**
FACT-M	0.0014	**0.8717**	5.7988
FACT-L	**0.0012**	<u>0.8710</u>	8.7674

Bold shows the best performance and <u>underline</u> shows the second-best. ↓: lower is better, ↑: higher is better.

D. Ablation Study (RQ3)

Ablation studies are conducted on the key components of FACT to assess the importance of each part. Specifically, we evaluated FACT with the absence of either condition variables $\boldsymbol{c}$ or AR. Besides, we verify different model sizes and different condition settings, checking their performance with efficiency.

The results, presented in Table VI, highlight the importance of both condition variables $\boldsymbol{c}$ and AR in achieving the best overall performance for the FACT model. Specifically, removing condition $\boldsymbol{c}$ (FACT w/o $\boldsymbol{c}$) leads to a significant decline in generation quality, as evidenced by the increased density error and length error, along with a substantial drop in the pattern score. While FACT-S achieves the smallest model size (4.3 MB), this reduction comes at the cost of compromised trajectory realism and compactness, making it unsuitable as a comprehensive solution. On the other hand, although FACT-AR reduces density error and length error compared to the original FACT since full trajectory data is utilized for training, the model size increases significantly (38.5 MB), leaving high space complexity. Besides, as the model is sensitive to the number of TDFormer blocks, performance differs when the blocks change. This tradeoff makes FACT-L less appealing when considering a balance between quality and efficiency. In contrast, the original FACT-M achieves the highest pattern score while maintaining a competitive balance across all other metrics, confirming its robustness as the most balanced approach for traffic trajectory generation.

Furthermore, we evaluate the performance of different condition settings. The simplification process reduces redundant features while maintaining similar performance. However, the

TABLE VII: Efficiency table for Top-3 Models.

Methods	**Diff-RNTraj**	**ControlTraj**	FACT	
			FACT	**FACT w/o AR**
Training time (seconds / epoch)	**623**	1197	<u>843</u>	2126
Inference time (seconds / batch)	**35**	126	<u>107</u>	318
# Params (million)	27.1774	8.1793	**5.7988**	<u>8.0518</u>

Bold shows the best performance, and <u>underline</u> shows the second-best.

real-world application of the condition faces challenges, as some key features are not provided for guidance, presenting further difficulties in achieving optimal results.

E. Efficiency Comparison (RQ4)

Efficiency is a critical consideration for real-world applications, where computational cost and scalability directly influence the feasibility of deployment. Time and space complexity is crucial for the application of the model in realistic scenarios. Therefore, we collect the training and generation time of each model, as well as their model size. Specifically, we choose **Diff-RNTraj**, **ControlTraj**, and **FACT** for comparison, which remarkably outperforms other baselines. We train each model by PyTorch framework with the Adam optimizer for 200 epochs and a batch size of 512. The details are shown below.

Based on the results in Table VII, the FACT model demonstrates significant advantages in the number of model parameters compared to Diff-RNTraj and ControlTraj. As one of the variants, FACT w/o AR gets the longest training time and inference time due to high time complexity without proper resampling. FACT achieves the lowest number of parameters and competitive training and inference time, making it the most lightweight and computationally efficient option. Although Diff-RNTraj boasts the fastest training and inference times, its significantly higher parameter count raises concerns about overall efficiency, particularly in resource-constrained environments. Although Diff-RNTraj has the fastest training and inference time, its significantly higher parameter count raises concerns about overall efficiency, particularly in resource-constrained environments. These results underscore the superiority of the FACT architecture in terms of resource efficiency. Overall, the FACT model offers a compelling balance of scalability and efficiency.

TABLE VIII: Utility test on traffic flow prediction task (Xi'an). Performance is shown as (original / generated).

Methods	ASTGCN	GWNet	DCRNN
RMSE	4.62 / 4.60	5.93 / 5.74	4.89 / 4.72
MAE	2.91 / 2.90	3.59 / 3.47	3.07 / 3.01

F. Real-world Database Application (RQ5)

The proposed framework is designed to generate data that is highly useful for downstream tasks, and substantially expand trajectory databases, enhancing their capacity to support real-world applications. To validate its utility, we selected the traffic flow prediction task [40], a critical task in Intelligent Transportation Systems (ITS), using data from Xi'an City. We trained prediction models, including ASTGCN [41], GWNet [42], and DCRNN [43], using the original data and evaluated them on both the original and generated datasets. The results, presented in Tables VIII, show that the generated data from FACT performs almost identically to the original data, confirming its utility in downstream tasks. This demonstrates that the generated trajectories can serve as a viable replacement for the original ones in future traffic-related tasks with equivalent performance, enabling data expansion for data-hungry trajectory systems. Furthermore, as shown in Table VI, the simplified condition $\boldsymbol{c}$ achieves performance comparable to $\boldsymbol{c}ref$, offering an efficient data compression solution for large-scale databases by requiring less storage than $\boldsymbol{c}ref$.

VI. Conclusion

This work proposes FACT, a high-fidelity and controllable trajectory generation framework designed to address the challenges of modeling complex spatial-temporal data. FACT leverages three key techniques: the TDFormer block for the efficient denoising process, well-defined condition variables for precise guidance, and the adaptive resampling strategy to reduce computational complexity. By integrating these modules, FACT provides a robust solution for generating high-quality trajectories while achieving fine-grained control over the generation process.

Extensive experiments conducted on datasets such as Chengdu, Xi'an, and Porto validate FACT's superiority over existing methods across key metrics like density accuracy and pattern fidelity, showing the superiority of TDFomer block over U-net architecture. Well-designed condition variables precisely guide the generation as well as provide a possible solution for data compression in large database scenarios. The effectiveness of the adaptive resampling strategy is highlighted in reducing model complexity without compromising utility in downstream tasks like traffic flow prediction.

While FACT has demonstrated its efficacy in trajectory generation, future work will focus on extending the framework to other types of spatial-temporal data to address a broader range of application domains. Additionally, we aim to optimize the model architecture for distributed databases to enhance scalability and efficiency in large-scale systems. These efforts will further establish FACT as a versatile and robust solution for real-world spatial-temporal data modeling challenges.

References

[1] S. Wang, Z. Bao, J. S. Culpepper, and G. Cong, "A Survey on Trajectory Data Management, analytics, and learning," *arXiv preprint arXiv:2003.11547*, 2020.

[2] Y. Zhu, Y. Ye, Y. Liu, and J. James, "Cross-Area Travel Time Uncertainty Estimation From Trajectory Data: A Federated Learning Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24 966–24 978, Dec. 2022.

[3] H. Shi, Q. Yao, Q. Guo, Y. Li, L. Zhang, J. Ye, Y. Li, and Y. Liu, "Predicting origin-destination flow via multi-perspective graph convolutional network," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 2020, pp. 1818–1821.

[4] A. G. Chekol and M. S. Fufa, "A survey on next location prediction techniques, applications, and challenges," *EURASIP Journal on Wireless Communications and Networking*, vol. 2022, no. 1, p. 29, Mar. 2022.

[5] B. Deng, B. Qu, P. Wang, D. Yang, B. Fankhauser, and P. Cudre-Mauroux, "Replay: Modeling time-varying temporal regularities of human mobility for location prediction over sparse trajectories," *arXiv preprint arXiv:2402.16310*, 2024.

[6] F. Simini, G. Barlacchi, M. Luca, and L. Pappalardo, "A deep gravity model for mobility flows generation," *Nature communications*, vol. 12, no. 1, p. 6576, 2021.

[7] J. Rao, S. Gao, Y. Kang, and Q. Huang, "LSTM-TrajGAN: A Deep Learning Approach to Trajectory Privacy Protection," in *11th International Conference on Geographic Information Science (GIScience 2021) - Part I*, ser. Leibniz International Proceedings in Informatics (LIPIcs), K. Janowicz and J. A. Verstegen, Eds., vol. 177. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, pp. 12:1–12:17. [Online]. Available: https://drops.dagstuhl.de/opus/volltexte/2020/13047

[8] T. Xia, X. Song, Z. Fan, H. Kanasugi, Q. Chen, R. Jiang, and R. Shibasaki, "DeepRailway: A Deep Learning System for Forecasting Railway Traffic," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, Apr. 2018, pp. 51–56.

[9] Y. Zhu, Y. Ye, S. Zhang, X. Zhao, and J. Yu, "Difftraj: Generating gps trajectory with diffusion probabilistic model," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 65 168–65 188.

[10] Y. Yuan, J. Ding, C. Shao, D. Jin, and Y. Li, "Spatio-temporal diffusion point processes," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, p. 3173–3184.

[11] Y. Zhu, J. J. Yu, X. Zhao, Q. Liu, Y. Ye, W. Chen, Z. Zhang, X. Wei, and Y. Liang, "Controltraj: Controllable trajectory generation with topology-constrained diffusion model," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 4676–4687.

[12] J. Feng, Z. Yang, F. Xu, H. Yu, M. Wang, and Y. Li, "Learning to simulate human mobility," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 3426–3433.

[13] Y. Yuan, J. Ding, H. Wang, D. Jin, and Y. Li, "Activity trajectory generation via modeling spatiotemporal dynamics," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4752–4762.

[14] Y. Wang, T. Zheng, S. Liu, Z. Feng, K. Chen, Y. Hao, and M. Song, "Spatiotemporal-augmented graph neural networks for human mobility simulation," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[15] X. Wang, X. Liu, Z. Lu, and H. Yang, "Large scale gps trajectory generation using map based on two stage gan," *Journal of Data Science*, vol. 19, no. 1, pp. 126–141, 2021.

[16] G. Xiong, Z. Li, M. Zhao, Y. Zhang, Q. Miao, Y. Lv, and F.-Y. Wang, "Trajsgan: A semantic-guiding adversarial network for urban trajectory generation," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 2, pp. 1733–1743, 2024.

[17] Z. Tao, W. Xu, and X. You, "Map2traj: Street map piloted zero-shot trajectory generation with diffusion model," *arXiv preprint arXiv:2407.19765*, 2024.

[18] M. Luca, G. Barlacchi, B. Lepri, and L. Pappalardo, "A survey on deep learning for human mobility," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–44, 2021.

[19] T. Wei, Y. Lin, S. Guo, Y. Lin, Y. Huang, C. Xiang, Y. Bai, and H. Wan, "Diff-rntraj: A structure-aware diffusion model for road network-constrained trajectory generation," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–15, 2024.

[20] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2256–2265.

[21] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[22] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[23] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[24] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.

[25] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.

[26] M. Mirza, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[27] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 336–345.

[28] H. P. Das, R. Tran, J. Singh, X. Yue, G. Tison, A. Sangiovanni-Vincentelli, and C. J. Spanos, "Conditional synthetic data generation for robust machine learning applications with limited pandemic data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, 2022, pp. 11 792–11 800.

[29] P. Cao, F. Zhou, Q. Song, and L. Yang, "Controllable generation with text-to-image diffusion models: A survey," *arXiv preprint arXiv:2403.04279*, 2024.

[30] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[31] A. Ulhaq and N. Akhtar, "Efficient diffusion models for vision: A survey," *arXiv preprint arXiv:2210.09292*, 2022.

[32] Y. Pu, Z. Xia, J. Guo, D. Han, Q. Li, D. Li, Y. Yuan, J. Li, Y. Han, S. Song *et al.*, "Efficient diffusion transformer with step-wise dynamic attention mediators," *arXiv preprint arXiv:2408.05710*, 2024.

[33] W. Zhao, Y. Han, J. Tang, K. Wang, Y. Song, G. Huang, F. Wang, and Y. You, "Dynamic diffusion transformer," *arXiv preprint arXiv:2410.03456*, 2024.

[34] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547 – 570, 2018.

[35] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 3–7 May. 2021.

[36] P. Goyal, "Accurate, large minibatch sg d: training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[37] W. Ding, M. Xu, and D. Zhao, "Cmts: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4314–4321.

[38] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *International Conference on Learning Representations*, 2020.

[39] Y. Du, Y. Hu, Z. Zhang, Z. Fang, L. Chen, B. Zheng, and Y. Gao, "Ldptrace: Locally differentially private trajectory synthesis," *Proceedings of the VLDB Endowment*, vol. 16, no. 8, pp. 1897–1909, 2023.

[40] R. Jiang, D. Yin, Z. Wang, Y. Wang, J. Deng, H. Liu, Z. Cai, J. Deng, X. Song, and R. Shibasaki, "Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction," in *Proceedings of the 30th ACM international conference on information & knowledge management*, 2021, pp. 4515–4525.

[41] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, “Attention based spatial-temporal graph convolutional networks for traffic flow forecasting,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 922–929.

[42] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Graph wavenet for deep spatial-temporal graph modeling,” *arXiv preprint arXiv:1906.00121*, 2019.

[43] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” *arXiv preprint arXiv:1707.01926*, 2017.