FACT: High-Fidelity and Controllable Trajectory Data Generation

Anonymous Author(s)

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-**F**idelity **A**nd **C**ontrollable **T**rajectory 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.

Keywords

Diffusion transformer, trajectory generation, controllable generation.

ACM Reference Format:

Anonymous Author(s). 2018. FACT: High-Fidelity and Controllable Trajectory Data Generation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

With the rapid proliferation of GPS-embedded devices and advanced data acquisition technologies, a substantial volume of trajectory data has been generated [39]. Trajectory data, comprising a sequence of spatial points that delineate vehicle movement over time, is extensively utilized in a diverse array of traffic-related applications owing to its rich spatiotemporal insights. These applications encompass travel time prediction [51], origin-destination analysis [32], and various location-based services [4], playing a crucial role in facilitating the advancement of Intelligent Transportation Systems (ITS). However, the acquisition of such extensive datasets presents considerable challenges, including privacy infringements, elevated procurement expenses, and stringent regulatory constraints on data dissemination [16, 17]. Furthermore, trajectory data exhibits significant inter-individual variability and sensitivity to spatiotemporal contexts, thereby impeding the accurate capture of vehicle mobility patterns [5, 18, 22]. This accentuates the imperative for an effective trajectory generation methodology that not only encapsulates the complicated data distributions and salient features but also enables controllable trajectory synthesis to underpin subsequent analytical endeavors.

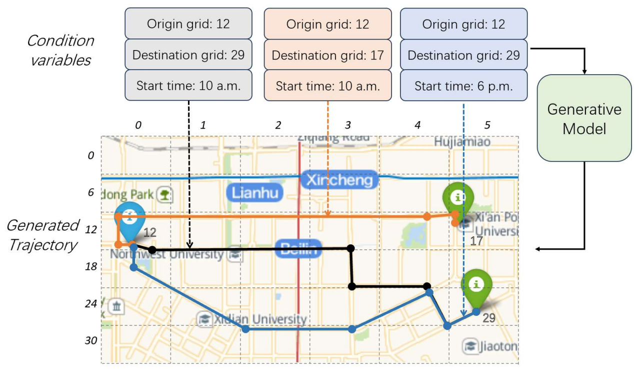

Figure 1: High-fidelity and controllable trajectory generation.

Several data generation methodologies leveraging deep learning techniques have been proposed to synthesize data that closely approximates the original data distribution and feature space [1, 23, 42, 44, 53]. By preserving the fidelity of the original data distribution, generative methods furnish a viable solution for data sharing, providing synthetic data as a substitute for raw data and mitigating privacy apprehensions[2]. Moreover, in comparison to other data modalities such as text and images, trajectory datasets are inherently sparse and often challenging to procure at a large scale [7]. By generating synthetic trajectories, generative models can significantly augment trajectory databases, thereby enhancing their utility in applications including mobility forecasting and spatiotemporal analysis [49].

In the pursuit of generating trajectories that capture the complexities inherent in real-world scenarios, researchers have explored a spectrum of approaches. Initial endeavors, grounded in rule-based or statistical paradigms, offered interpretability yet encountered limitations in capturing the intricate spatiotemporal dynamics of human mobility [33]. Subsequently, deep learning methodologies, such as Generative Adversarial Networks (GANs) [29], Variational Autoencoders (VAEs) [45], and Diffusion Models [47, 52, 53], have

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

been increasingly adopted to model complex trajectory distributions. Despite the advancements afforded by these techniques, current trajectory modeling approaches commonly prioritize the overall trends in vehicle transitions. This is often achieved through the simplification of trajectories into coarse-grained representations, such as grid cells [12, 41, 48], or by transforming them into image-based formats [37, 40, 46]. Consequently, this emphasis on simplification can lead to limitations in the granularity of the generated trajectories [24]. Furthermore, current methodologies, including Diff-RNTraj [42] and DiffTraj [52], exhibit limitations in generating specific trajectory patterns, particularly those characterized by irregularity or sparsity in spatial distribution. Moreover, these approaches often fall short of meeting the requirements of application-driven scenarios, such as the demand for fine-grained spatiotemporal control or the accommodation of heterogeneous, user-specified conditions during the trajectory synthesis process. Challenges persist in generating fine-grained, condition-specific trajectory data, and can be summarized as follows:

- **Fidelity**: Generating high-fidelity trajectories is difficult due to complex spatiotemporal features and the necessity of incorporating road network topology. Current methods typically require pre-training for topology encoding and output rectification.
- **Controllability**: Existing models primarily focus on group mobility simulation without precise individual control or task/region-specific guidance. The absence of explicit trajectory descriptors further complicates controllable generation for applications.
- **Efficiency**: General diffusion models, while offering high-quality generation, suffer from long training and inference times, posing efficiency concerns. Balancing model performance and computational efficiency remains a key challenge.

To address these aforementioned limitations, we introduce **FACT**, a **F**idelity-**A**nd-**C**ontrollable generative model that leverages diffusion models for trajectory data generation, as depicted in Figure 1. FACT incorporates a Trajectory-Denoising transFormer architecture, termed **TDFormer**, specifically designed for generating high-fidelity trajectory data. Furthermore, condition variables are meticulously designed to encompass multiple features to accurately represent trajectory characteristics, effectively guiding the training procedure and facilitating the generation of controllable outputs while minimizing redundant features to enhance efficiency. To mitigate the computational overhead associated with transformer architectures 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, in conjunction with condition variables, is subsequently integrated into the TDFormer to synthesize geographic trajectories from random Gaussian noise. Extensive experimental evaluations conducted on three real-world datasets corroborate the efficacy of FACT. The results demonstrate FACT's capability 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:

- This paper introduces the FACT framework, a novel approach predicated on the proposed TDFormer architecture, for generating high-fidelity trajectories. Synthesized trajectories from FACT preserve the underlying data distribution of the original dataset, thereby accurately representing real-world traffic dynamics and facilitating a range of downstream tasks.
- We present a condition variable design specifically tailored for trajectory data, which enables controllable trajectory generation through an efficient representation of trajectory characteristics. Additionally, we introduce an adaptive resampling strategy designed to guide both the training and generation processes, thereby enhancing model efficiency and operational flexibility.
- We empirically validate FACT utilizing three real-world datasets. The results demonstrate superior performance in generating high-fidelity trajectory data relative to baseline methodologies. Furthermore, evaluations on downstream tasks highlight the efficacy of the generated trajectories.

The remainder of this paper is structured as follows: Section 2 reviews related work on generative models, conditional generation, and model efficiency. Sections 3 and 4 present the problem definition and the proposed methodology, respectively. Section 5 details the experimental setup and compares the performance of FACT with state-of-the-art models. Finally, Section 6 concludes the paper and outlines future research directions.

2 Related Work

In this section, we present a brief review of prior research pertaining to generative models, conditional generation techniques, and considerations of model efficiency. Our discussion emphasizes diffusion models, owing to their outstanding performance when compared to other contemporary generative modeling paradigms.

2.1 Diffusion Model for Trajectory Generation

As a state-of-the-art data generation technique, the diffusion model showcases its robust capabilities in generating data [14, 34, 36]. 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) [35] 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 [26]. Additionally, numerous studies have explored spatial-temporal generation using the diffusion model. For example, DiffTraj [52] employs the diffusion model with a U-Net architecture to generate trajectory data. ControlTraj [53] extends this approach by incorporating topology constraints to produce higher-quality data. Diff-RNTraj [42] 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 [27].

2.2 Conditional Generation

Conditional data generation has become a crucial area in machine learning, especially through the use of Conditional Generative Adversarial Networks (CGAN) [25] and Conditional Variational Autoencoders (CVAE) [21]. 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 [6]. Furthermore, diffusion models can provide guidance for the given conditions [3]. Condition variables are embedded and concatenated with hidden variables during training [52], or they can interact with the model training through mechanisms like cross-attention [27, 30]. This allows the model to be directed by the condition variables, enabling the desired generation.

2.3 Model Efficiency

Although the diffusion model outperforms other methods with higher generation quality, there remains a challenge in balancing computational cost and model performance [38]. 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 [28, 50]. Therefore, optimizations are necessary for both training and inference to make the model more efficient in real-world applications.

3 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.

3.1 Problem Definition

Definition 1 (GPS Trajectory). *Formally, a GPS trajectory $\mathcal{P}$ is defined as a temporally ordered sequence of GPS points, $\mathcal{P} = \{p_1, \dots, p_m\}$. Each point $p_i \in \mathcal{P}$ is represented as a triplet $p_i = \langle lon_i, lat_i, t_i \rangle$, where lon_i and lat_i denote the longitude and latitude, respectively, at time step t_i. The triplet p_i thus encapsulates the spatial coordinates and the timestamp for the i-th point in the trajectory.*

Definition 2 (Traffic road network). *The traffic road network, representing the topological structure of a city, is formally defined as a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. In this graph, $\mathcal{V}$ represents the set of vertices, corresponding to intersections or junctions within the road network, and $\mathcal{E}$ denotes the set of directed edges, representing the road segments connecting these intersections. Road topology information is publicly accessible from resources such as OpenStreetMap*[1].

Definition 3 (Trajectory condition variable). *Let $\mathbf{c} = \{f_1, f_2, \dots\}$ denote the condition variable associated with a given trajectory, where each element f_i represents the i-th trajectory feature. This vector encapsulates a set of salient features characterizing the trajectory, designed to be readily discernible by the generative model. Furthermore, the condition variable is constructed to efficiently represent the conditions influencing the trajectory, ensuring feature distinctiveness and minimizing redundancy.*

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

Problem 1 (Trajectory Generation). *Given a dataset of original GPS trajectories $\mathcal{T} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n\}$, where each $\mathcal{P}_i = \{p_1^i, \dots, p_m^i\}$ constitutes an individual trajectory, the objective is to train a generative model $\mathbf{G}$. This model is tasked with generating a set of synthetic trajectories $\mathcal{T}_k = \mathbf{G}(\mathbf{C}_k)$, where $\mathbf{C}_k$ represents the condition set for the generated trajectories and k denotes the cardinality of the generated set. The generated trajectories are required to exhibit a data distribution statistically similar to that of $\mathcal{T}$, demonstrating high fidelity with respect to the underlying road network topology $\mathcal{G}$ of the corresponding city. Moreover, the generation process must be effectively controlled by the provided condition set $\mathbf{C}_k$. Finally, the computational demands of the generation process must be tenable for practical real-world applications.*

3.2 Diffusion Probabilistic Model

The diffusion probabilistic model, frequently denoted as the diffusion model, has emerged as a potent paradigm for the generation of high-quality data, spanning modalities such as images, text, and audio [11, 15, 19, 20, 31, 43]. This versatility and performance efficacy render diffusion models superior to alternative generative approaches like GAN and VAE [14, 35, 36]. At its fundamental principle, the diffusion model operates through two stages: the forward diffusion process and the reverse denoising process. The forward process entails the iterative and gradual addition of random noise to the original data, whereas the reverse process involves training a model to learn the inverse transformation, effectively reconstructing the original data from its progressively noised counterpart. These processes can be mathematically formulated as follows:

Forward process. Given original data denoted as X_0, the forward diffusion process is characterized by the progressive addition of Gaussian noise over T discrete steps. This process is formally structured as a Markov chain, defined as:

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

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

where $\mathbf{I}$ represents the identity matrix and $\beta_t \in (0,1)_{t=1}^{T}$ is a sequence of variances. To enable differentiability and facilitate gradient-based optimization, the reparameterization trick is employed [14]. This technique allows for the expression of X_t in terms of X_0 and a noise term, specifically $X_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$ and $\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$.

Reverse process. In the reverse process, the generative model is trained to approximate the reverse Markov chain and reconstruct the original data distribution from a state of pure noise. The initial state of the reverse process is given by sampling from a standard Gaussian distribution, $X_t \sim \mathcal{N}(0, \mathbf{I})$. Analogous to the forward

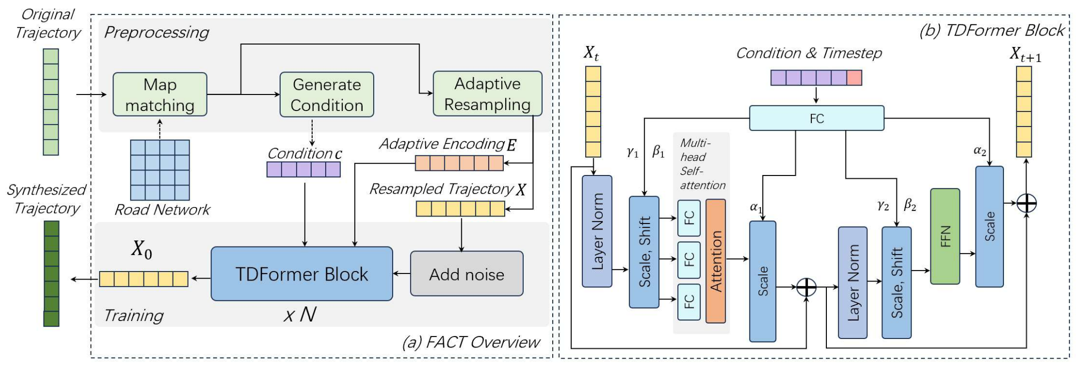

Figure 2: Network structure of FACT. (a) FACT Overview. (b) TDFormer Block.

process, the reverse process is also formulated 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 the mean and variance functions, respectively, parameterized by θ. Following the reparameterization technique [14], for any $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ $(t > 1)$ and $\tilde{\beta}_1 = \beta_1$, the parameters μ_θ and σ_θ are specified 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:

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

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

3.3 Diffusion Transformer

The Diffusion Transformer (DiT) [27] integrates diffusion models with Vision Transformers (ViT) [9] for high-fidelity generation, particularly in image synthesis. It processes data as embedded tokens and incorporates temporal conditioning through trainable timestep embeddings, effectively handling temporal dependencies. Following the standard diffusion framework, DiT applies forward noise addition and reverse denoising via transformers. Utilizing a frozen pre-trained encoder E to map inputs into a latent space, DiT trains the diffusion process on $z = E(X)$, subsequently generating data by sampling z', which is decoded by a pre-trained decoder D to obtain $X' = D(z')$. DiT achieves state-of-the-art performance in high-resolution image generation, setting a benchmark for complex generative tasks.

4 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.

4.1 FACT Overview

The FACT framework, as illustrated in Figure 2, is designed to generate high-fidelity, controllable trajectories by systematically transforming raw inputs through a structured pipeline. The process begins with map matching, which ensures that trajectories are accurately aligned with the road network, reducing noise and inconsistencies in the raw data. Next, condition generation extracts key constraints, such as spatial regions, temporal constraints, or traffic conditions, to guide the synthesis process, enabling precise control over the generated trajectories. To further enhance consistency, adaptive resampling standardizes trajectory representations, ensuring uniform input sequences that improve model stability and learning efficiency. These refined inputs are then encoded to capture critical spatial-temporal features before being passed to the TDFormer block, which effectively models complex dependencies across time and space. By iteratively applying denoising and refinement, FACT ensures that the generated trajectories not only replicate real-world movement patterns but also strictly adhere to the given conditions, making them highly reliable for downstream applications.

4.2 TDFormer Block

As outlined in Section 3, FACT is designed as a denoising framework, aiming to develop a neural network architecture that precisely estimates and removes the noise component $\epsilon_\theta(X_t,t)$ at each diffusion timestep t. To achieve this objective, we introduce the TDFormer block, a novel architectural element that integrates adaptive Trajectory Layer Normalization (adaTLN) with Multi-Head Self-Attention (MHSA) mechanisms. This integration facilitates precise modeling of trajectory-specific conditions and inherent spatiotemporal characteristics. The adaTLN mechanism is specifically engineered to incorporate conditional guidance by dynamically modulating normalization parameters according to the trajectory context. Concurrently, MHSA serves to capture intra-sample data correlations within trajectory sequences.

In implementation, the adaTLN mechanism is realized through a scale and shift module. This module employs learnable scale ($\boldsymbol{\gamma}$) and shift ($\boldsymbol{\beta}$) parameters, which are conditioned on the diffusion timestep t and the condition variable c. Furthermore, we incorporate dimension-wise scaling parameters, denoted as $\boldsymbol{\alpha}$ [27], immediately prior to the residual connections within the TDFormer block. These parameters modulate the output signal of each module, allowing for dynamic adjustment of their contribution to the aggregated output. Moreover, zero-initialization of the final batch normalization scale factor $\boldsymbol{\gamma}$ in each TDFormer block is employed to expedite large-scale training procedures, particularly in supervised learning scenarios [13]. The computational flow of these parameter interactions is formally described by the equations below:

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

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

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

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. As illustrated in Figure 2, the outcome of the TDFormer block is computed as $\boldsymbol{H}^{i+1} = \text{TDFormer}(\boldsymbol{H}^i)$, 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 processing mechanism allows the model to progressively refine and enhance trajectory predictions at each successive layer, inherently considering both spatiotemporal dependencies and externally provided conditional guidance. The details of TDFormer block are presented in Appendix A.

4.3 Trajectory Condition Design

Unlike images, trajectory data lacks explicit categorical labels for conditional guidance, making most generative models inherently unconditional. Building on previous studies [52, 53], FACT refines condition variables by eliminating redundancy and enhancing efficiency. This streamlined approach not only improves scalability and practicality but also achieves or exceeds the performance of existing methods in condition-guided generation.

Within trajectory analysis, condition variables can include *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), initially defined as:

$$\mathbf{c}_{\text{all}} = \{dt, td, tt, tl, ad, as, oID, dID\}. \tag{11}$$

From this, we derive a simplified, efficient version, retaining key discrete spatial-temporal features and crucial trajectory guidance features (td, tt, tl). The remaining features are either linearly derivable from the selected subset or exert negligible influence on the generation outcomes, a hypothesis that is empirically validated in Section 5.3, resulting in:

$$\mathbf{c}_{\text{less}} = \{dt, td, tt, tl, oID, dID\}. \tag{12}$$

However, in practical trajectory data generation scenarios, the granular details of trajectories intended for synthesis are typically not known *a priori*. This inherent limitation renders trajectory features that are contingent upon completed trajectories, such as total distance or total length, inaccessible prior to the generation process. We therefore further condense the condition variable

$$\mathbf{c} = \{dt, oID, dID\}, \tag{13}$$

which exclusively incorporates departure time, origin grid ID, and destination grid ID. These features are characteristically and readily obtainable in practical, real-world deployments. A more detailed evaluation of the guidance efficacy of these diverse condition variable formulations is subsequently provided in Section 5.3. The details of trajectory conditions are presented in Appendix B.

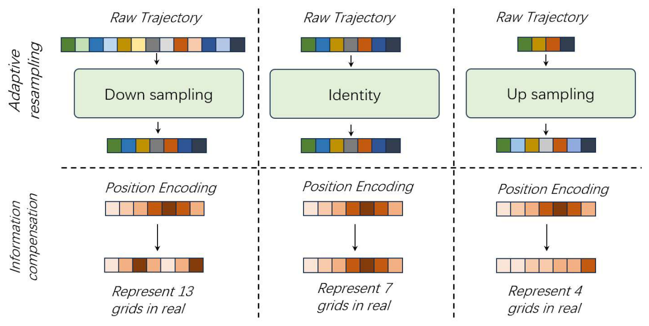

Figure 3: Adaptive resampling.

4.4 Adaptive Resampling Strategy

As depicted in Figure 3, the adaptive resampling module comprises two key components: resampling and information compensation.

4.4.1 Adaptive Resampling. Transformer complexity increases with input sequence length. To mitigate this, we resample trajectories of varying lengths to a fixed length $\boldsymbol{L}$ using linear interpolation:

$$\hat{\mathcal{P}}' = \text{AR}(\hat{\mathcal{P}}, \boldsymbol{L}), \tag{14}$$

where $\hat{\mathcal{P}}'$ is the resampled trajectory, $\hat{\mathcal{P}}$ is the original map-matched trajectory, and $\boldsymbol{L}$ is the target length.

4.4.2 Information Compensation. While adaptive resampling effectively reduces computational overhead, it inherently introduces the potential for information loss, especially in long-distance trajectories where intermediate points are omitted. To mitigate this, we employ an Information Compensation (IC) mechanism by updating positional encodings. We adjust the positional encoding to reflect the resampled trajectory and maintain temporal accuracy. The updated positional encoding is calculated as:

$$PE_{(pos_L,2i)} = \sin\left(\frac{pos_L}{10000^{2i/d_{\text{model}}}}\right) = \sin\left(\frac{L}{l_0} \cdot \frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{15}$$

$$PE_{(pos_L,2i+1)} = \cos\left(\frac{pos_L}{10000^{2i/d_{\text{model}}}}\right) = \cos\left(\frac{L}{l_0} \cdot \frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{16}$$

where l_0 is the original trajectory length, pos is the original position, and $pos_L = pos * l_0/L$ is the adjusted position for the fixed length $\boldsymbol{L}$. The scaling factor l_0/L ensures positional encodings accurately represent time intervals in the resampled trajectory, preserving temporal information and improving model processing of resampled data.

4.5 Trajectory Generation

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

4.5.1 Training. 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:

$$\min_{\theta} \mathcal{L}(\theta) = \min_{\theta} \mathbb{E}_{\boldsymbol{c},t,X_0\sim q} \left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_{\theta}\left(X_t,\ t,\ \boldsymbol{c}\right) \right\|_2^2, \tag{17}$$

4.5.2 Generation. During sampling, we start with $X_T \sim \mathcal{N}(0, \mathbf{I})$, a Gaussian noise initialization. The trajectory is then recursively sampled via the learned reverse process: $X_{t-1} \sim p_\theta(X_{t-1}|X_t)$, modeling the transition from X_t to X_{t-1}. The reparameterization trick ensures that the model can generate samples by backpropagating through the noise addition process.

5 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? How does each feature in the condition variable work?
- **RQ3**: How does each module in FACT contribute to the overall generation performance?
- **RQ4**: How does FACT perform with model scaling and dataset scaling?

5.1 Experimental Settings

5.1.1 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, encapsulating diverse urban mobility dynamics. A detailed summary is available in Appendix D.1 for reference.

5.1.2 Baselines. We compare the proposed FACT with state-of-the-art generative methods, including Conditional VAE (CVAE) [8], Conditional GAN (CGAN) [25], and diffusion models, DiffWave [20], DiffTraj [52], Diff-RNTraj [42] and ControlTraj [53]. The detailed description are presented in Appendix D.2 for reference.

5.1.3 Evaluation Metrics. We follow the methodology from previous work [10, 53] 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. We randomly generate 9,000 trajectories from each model and calculate their metrics. Further details are presented in Appendix D.3.

5.1.4 Implement Details. In FACT, we conduct model hyperparameter settings and training settings. See Appendix C for details.

5.2 Overall Performance (RQ1)

FACT surpasses all baselines across three datasets, as shown in Table 1. In Chengdu, it reduces density and length errors by 80.65% and 54.64%, respectively, and improves the pattern score by 22.16%. In Porto, it lowers density and length errors by 53.06% and 35.42%, with a 19.69% pattern score gain. These results highlight the TDFormer block's superiority over U-Net. While Diff-RNTraj enforces road constraints, it struggles with increasing road segment complexity. As illustrated in Figure 4, FACT performs comparably to other baselines in dense regions of Xi'an while surpassing them in sparse areas, ultimately achieving superior overall performance. Full visualization is presented in Appendix D.4.

Performance variations across datasets indicate the influence of road network complexity. In complex road networks like Chengdu, FACT's modeling capabilities are crucial, whereas in road networks like Porto, its relative advantage is smaller but still notable for capturing trajectory patterns.

FACT strikes an optimal balance between efficiency and effectiveness. We compare training and inference times for Diff-RNTraj, ControlTraj, and FACT, using a batch size of 256. Although Diff-RNTraj has the fastest training (623s/epoch) and inference (35s/batch) times, its performance is inferior, reflecting the trade-off between efficiency and effectiveness. FACT, utilizing simplified conditions and adaptive resampling, achieves strong performance with reasonable overhead (843s/epoch, 270s/batch), outperforming ControlTraj (1197s/epoch, 326s/batch). FACT delivers the best performance and an acceptable trade-off between time and quality, demonstrating its overall superiority. Further details on efficiency are in Appendix D.5.

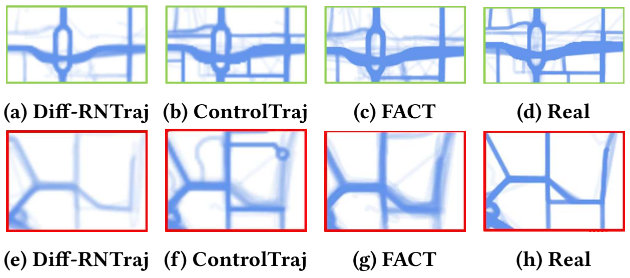
(a) Diff-RNTraj (b) ControlTraj (c) FACT (d) Real
(e) Diff-RNTraj (f) ControlTraj (g) FACT (h) Real

Figure 4: Visualization of generated trajectory dataset in Xi'an City. Red boxes show the sparse area. Green boxes show the dense area.

5.3 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

Table 1: 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**
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**
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**

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

Table 2: Controllable generation study.↓: lower is better, ↑: higher is better.

Methods	OD Accuracy (↑)	TD error (↓)	AS error (↓)
CVAE	55.30% / 55.62%	1227 m	3.480 m/s
CGAN	55.76% / 58.03%	1203 m	3.368 m/s
DiffTraj	91.59% / 91.41%	139.1 m	0.2179 m/s
ControlTraj	93.22% / 93.62%	86.10 m	0.1590 m/s
FACT	**94.24% / 94.69%**	**82.60m**	**0.1384m/s**

the generated results. 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: OD (Origin grid and Destination grid) Accuracy, TD (Total Distance) error, and AS (Average Speed) error.

The results in Table 2 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.

We further conduct a single-feature analysis on the condition variables to evaluate the controllability of each feature, as defined in Equation 12. As shown in Table 3, we employ mutual information to quantify the relationship between generated trajectories and condition guidance, where higher mutual information values indicate stronger dependencies.

The results reveal that the highest mutual information values correspond to the oID (4.90) and dID (4.88) features, suggesting that the model is particularly sensitive to these inputs when generating trajectories. These features play a pivotal role in shaping the overall spatial structure, ensuring the generated trajectories align with real-world movement patterns. Conversely, certain features, such as

Table 3: Single feature study.

Variable	td	tt	tl	oID	dID
Mutual Information	0.86	1.31	0.77	4.90	4.88

total distance and total length, exhibit weaker influence, indicating potential areas for optimization and further refinement.

5.4 Ablation Study (RQ3)

The ablation studies delve into the contributions of key components in FACT, aiming to understand their impact on model performance. Table 4 highlights that both the condition variables $\boldsymbol{c}$ and the adaptive resampling (**AR**) module are critical for achieving optimal trajectory generation quality. The analysis also evaluates alternative transformer-based architectures and condition fusion strategies to discern the significance of the TDFormer block.

The absence of condition variables (**FACT w/o c**) leads to a pronounced drop in performance, evidenced by higher density errors and a notable reduction in the pattern score. This underscores the pivotal role of $\boldsymbol{c}$ in providing essential spatial-temporal guidance to the generative process. The degradation in metrics indicates

Table 4: Ablation study on FACT.

Metrics	Xi'an		
	Density (↓)	Pattern (↑)	# Params (million)
FACT w/o AR	<u>0.0015</u>	<u>0.8706</u>	8.0518
FACT w/o $\boldsymbol{c}$	0.0049	0.8570	5.4780
FACT-$\boldsymbol{c}_{\text{all}}$	**0.0014**	**0.8717**	5.7990
FACT-$\boldsymbol{c}_{\text{less}}$	**0.0014**	**0.8717**	5.7988
FACT-$\boldsymbol{c}$	0.0037	0.8689	<u>5.7983</u>
FACT-Cross Attention	0.0029	0.8652	5.8020
FACT-In-Context	0.0037	0.8601	5.4487
FACT-adaTLN	0.0027	0.8659	5.7988
FACT	**0.0014**	**0.8717**	5.7988

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

that condition variables are indispensable for preserving trajectory fidelity and ensuring consistency with the input guidance. Excluding the adaptive resampling module (**FACT w/o AR**) results in a slightly higher density error compared to the full model but achieves the second-best overall performance. However, this comes at a significant cost to model efficiency, as indicated by a 38.5% increase in parameter count. This highlights the trade-off between maintaining model compactness and achieving optimal trajectory generation quality. The **AR** module's contribution lies in efficiently handling trajectory data while minimizing computational overhead.

Further analysis of different condition settings reveals that simplifying condition variables by reducing redundant features maintains comparable performance. However, real-world applications pose challenges, as the absence of certain key features for guidance can hinder the model's ability to achieve optimal results.

The TDFormer block outperforms alternative transformer variants, including Cross Attention (**FACT-Cross Attention**) and In-Context Conditioning (**FACT-In-Context**). Both alternatives exhibit higher density errors and lower pattern scores, reflecting their limitations in effectively fusing condition information. These results underscore the superiority of TDFormer in leveraging condition variables for accurate and high-quality trajectory generation.

The **FACT-adaTLN** variant, which employs an adaptive trajectory layer normalization without zero-initialization, exhibits slightly better density error compared to other ablated models but falls short with FACT which applies zero-initialization. This demonstrates that while adaTLN aids feature adaptation, the absence of zero-initialization compromises optimization stability and limits the model's ability to capture intricate trajectory patterns effectively.

5.5 Model Scaling and Dataset scaling (RQ4)

The analysis of dataset scaling and model scaling provides insights into how data quantity and model capacity affect performance. In real-world scenarios, trajectory data is often scarce. As shown in Table 5, expanding the dataset from 25% to 100% leads to substantial performance gains across all metrics, highlighting the pivotal role of data volume in enhancing model effectiveness. More notably, even when trained on just 50% of the dataset, FACT generates trajectories that closely resemble real-world movements, demonstrating its robustness in data-limited settings, a scenario of practical importance.

Furthermore, we investigate model scaling, recognizing that increasing model depth enhances performance, yet determining the optimal depth for practical applications remains essential. In FACT, we employ a sequence of N TDFormer blocks and each operates

Table 5: Dataset scaling study on FACT.

Dataset ratio	Xi'an		
	Density (↓)	Length (↓)	Pattern (↑)
25%	0.0054	0.0167	0.8201
50%	0.0020	0.0074	0.8699
75%	0.0016	0.0072	0.8710
100%	0.0014	0.0069	0.8717

↓: lower is better, ↑: higher is better.

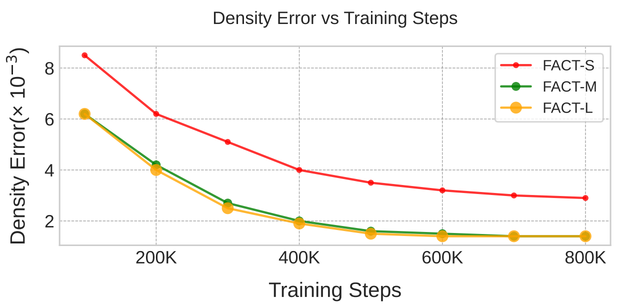

Figure 5: Model scaling.

with a hidden dimension size d. Following the approach in DiT [27], we use standard transformer configurations that jointly scale N, d, and the number of attention heads. Specifically, we use three 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 Appendix C.

As illustrated in Figure 5, which compares different model sizes (e.g., FACT-S/M/L), the results emphasize the crucial role of model capacity in performance improvement. Larger models, such as FACT-M, consistently surpass their smaller counterparts at all training stages, as indicated by the reduction in density error. This suggests that scaling up the model strengthens its learning and trajectory generation capabilities. However, as the model size continues to increase, it eventually reaches a saturation point, exemplified by FACT-L, beyond which further expansion provides no additional performance benefits, making excessive scaling unnecessary.

Together, these findings highlight the synergistic effect of dataset scaling and model scaling: larger datasets provide richer information for learning, while larger models are better equipped to leverage this information to achieve superior performance.

6 Conclusion

This work proposes FACT, a high-fidelity and controllable trajectory generation framework, which leverages three key techniques: the TDFormer block with adaTLN-Zero mechanism for the efficient denoising process, well-defined condition variables for precise guidance, and the adaptive resampling strategy to reduce computational complexity. Trajectory data is map-matched, and adaptively resampled before adding noise for training. FACT is trained to reconstruct trajectory from noisy data through its conditional guidance, and effectively denoises the random Gaussian noise by leveraging the condition variable to generate trajectory data. Based on the rigorous process, FACT provides a robust solution for generating high-quality trajectories with fine-grained control over the process. Extensive experiments on real-world trajectory datasets validate FACT's superiority over existing methods, showing the superiority of TDFomer block over U-net architecture. Well-designed condition variables precisely guide the generation and achieve efficient controllability. Additionally, the scalability of FACT is demonstrated across different datasets and model sizes. Future work will focus on extending the framework to other types of spatial-temporal data to address a broader range of application domains.

References

[1] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. 2022. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 11 (2022), 7327–7347. https://doi.org/10.1109/TPAMI.2021.3116668

[2] Zhipeng Cai, Zuobin Xiong, Honghui Xu, Peng Wang, Wei Li, and Yi Pan. 2021. Generative Adversarial Networks: A Survey Toward Private and Secure Applications. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–38.

[3] Pu Cao, Feng Zhou, Qing Song, and Lu Yang. 2024. Controllable Generation with Text-to-Image Diffusion Models: A Survey. *arXiv preprint arXiv:2403.04279* (2024).

[4] Ayele Gobezie Chekol and Marta Sintayehu Fufa. 2022. A survey on next location prediction techniques, applications, and challenges. *EURASIP Journal on Wireless Communications and Networking* 2022, 1 (Mar. 2022), 29.

[5] Xiaobo Chen, Huanjia Zhang, Feng Zhao, Yu Hu, Chenkai Tan, and Jian Yang. 2022. Intention-Aware Vehicle Trajectory Prediction Based on Spatial-Temporal Dynamic Attention Network for Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 23, 10 (2022), 19471–19483.

[6] Hari Prasanna Das, Ryan Tran, Japjot Singh, Xiangyu Yue, Geoffrey Tison, Alberto Sangiovanni-Vincentelli, and Costas J Spanos. 2022. Conditional Synthetic Data Generation for Robust Machine Learning Applications with Limited Pandemic Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 11792–11800.

[7] Bangchao Deng, Bingqing Qu, Pengyang Wang, Dingqi Yang, Benjamin Fankhauser, and Philippe Cudre-Mauroux. 2024. REPLAY: Modeling Time-Varying Temporal Regularities of Human Mobility for Location Prediction over Sparse Trajectories. *arXiv preprint arXiv:2402.16310* (2024).

[8] Wenhao Ding, Mengdi Xu, and Ding Zhao. 2020. CMTS: A Conditional Multiple Trajectory Synthesizer for Generating Safety-Critical Driving Scenarios. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 4314–4321. https://doi.org/10.1109/ICRA40945.2020.9197145

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[10] Yuntao Du, Yujia Hu, Zhikun Zhang, Ziquan Fang, Lu Chen, Baihua Zheng, and Yunjun Gao. 2023. LDPTrace: Locally Differentially Private Trajectory Synthesis. *Proceedings of the VLDB Endowment* 16, 8 (2023), 1897–1909.

[11] Ben Fei, Zhaoyang Lyu, Liang Pan, Junzhe Zhang, Weidong Yang, Tianyue Luo, Bo Zhang, and Bo Dai. 2023. Generative Diffusion Prior for Unified Image Restoration and Enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9935–9946.

[12] Jie Feng, Zeyu Yang, Fengli Xu, Haisu Yu, Mudan Wang, and Yong Li. 2020. Learning to Simulate Human Mobility. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3426–3433.

[13] P Goyal. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv preprint arXiv:1706.02677* (2017).

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

[15] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. 2023. Make-An-Audio: Text-To-Audio Generation with Prompt-Enhanced Diffusion Models. In *International Conference on Machine Learning*. PMLR, 13916–13932.

[16] Bin Jiang, Jianqiang Li, Guanghui Yue, and Houbing Song. 2021. Differential Privacy for Industrial Internet of Things: Opportunities, Applications, and Challenges. *IEEE Internet of Things Journal* 8, 13 (2021), 10430–10451.

[17] Hongbo Jiang, Jie Li, Ping Zhao, Fanzi Zeng, Zhu Xiao, and Arun Iyengar. 2021. Location Privacy-preserving Mechanisms in Location-based Services: A Comprehensive Survey. *ACM Computing Surveys (CSUR)* 54, 1 (2021), 1–36.

[18] Jiawei Jiang, Dayan Pan, Houxing Ren, Xiaohan Jiang, Chao Li, and Jingyuan Wang. 2023. Self-supervised Trajectory Representation Learning with Temporal Regularities and Travel Semantics. In *2023 IEEE 39th international conference on data engineering (ICDE)*. IEEE, 843–855.

[19] Gwanghyun Kim and Se Young Chun. 2023. DATID-3D: Diversity-Preserved Domain Adaptation Using Text-to-Image Diffusion for 3D Generative Model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 14203–14213.

[20] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2020. DiffWave: A Versatile Diffusion Model for Audio Synthesis. In *International Conference on Learning Representations*.

[21] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. 2017. DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 336–345.

[22] Mingqian Li, Panrong Tong, Mo Li, Zhongming Jin, Jianqiang Huang, and Xian-Sheng Hua. 2021. Traffic Flow Prediction with Vehicle Trajectories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 294–302.

[23] Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. 2018. ST-GAN: Spatial Transformer Generative Adversarial Networks for Image Compositing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Massimiliano Luca, Gianni Barlacchi, Bruno Lepri, and Luca Pappalardo. 2021. A Survey on Deep Learning for Human Mobility. *ACM Computing Surveys (CSUR)* 55, 1 (2021), 1–44.

[25] Mehdi Mirza. 2014. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784* (2014).

[26] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved Denoising Diffusion Probabilistic Models. In *International conference on machine learning*. PMLR, 8162–8171.

[27] William Peebles and Saining Xie. 2023. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4195–4205.

[28] Yifan Pu, Zhuofan Xia, Jiayi Guo, Dongchen Han, Qixiu Li, Duo Li, Yuhui Yuan, Ji Li, Yizeng Han, Shiji Song, et al. 2024. Efficient Diffusion Transformer with Step-wise Dynamic Attention Mediators. *arXiv preprint arXiv:2408.05710* (2024).

[29] Jinmeng Rao, Song Gao, Yuhao Kang, and Qunying Huang. 2020. LSTM-TrajGAN: A Deep Learning Approach to Trajectory Privacy Protection. In *11th International Conference on Geographic Information Science (GIScience 2021) - Part I (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 177)*, Krzysztof Janowicz and Judith A. Verstegen (Eds.). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 12:1–12:17. https://doi.org/10.4230/LIPIcs.GIScience.2021.I.12

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

[31] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. 2022. Palette: Image-to-Image Diffusion Models. In *ACM SIGGRAPH 2022 conference proceedings*. 1–10.

[32] Hongzhi Shi, Quanming Yao, Qi Guo, Yaguang Li, Lingyu Zhang, Jieping Ye, Yong Li, and Yan Liu. 2020. Predicting Origin-Destination Flow via Multi-Perspective Graph Convolutional Network. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 1818–1821. https://doi.org/10.1109/ICDE48307.2020.00178

[33] Filippo Simini, Gianni Barlacchi, Massimilano Luca, and Luca Pappalardo. 2021. A Deep Gravity model for mobility flows generation. *Nature communications* 12, 1 (2021), 6576.

[34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*. PMLR, 2256–2265.

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising Diffusion Implicit Models. *arXiv preprint arXiv:2010.02502* (2020).

[36] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-Based Generative Modeling through Stochastic Differential Equations. *arXiv preprint arXiv:2011.13456* (2020).

[37] Zhenyu Tao, Wei Xu, and Xiaohu You. 2024. Map2Traj: Street Map Piloted Zero-shot Trajectory Generation with Diffusion Model. *arXiv preprint arXiv:2407.19765* (2024).

[38] Anwaar Ulhaq and Naveed Akhtar. 2022. Efficient Diffusion Models for Vision: A Survey. *arXiv preprint arXiv:2210.09292* (2022).

[39] Sheng Wang, Zhifeng Bao, J Shane Culpepper, and Gao Cong. 2020. A Survey on Trajectory Data Management, Analytics, and Learning. *arXiv preprint arXiv:2003.11547* (2020).

[40] Xingrui Wang, Xinyu Liu, Ziteng Lu, and Hanfang Yang. 2021. Large Scale GPS Trajectory Generation Using Map Based on Two Stage GAN. *Journal of Data Science* 19, 1 (2021), 126–141.

[41] Yu Wang, Tongya Zheng, Shunyu Liu, Zunlei Feng, Kaixuan Chen, Yunzhi Hao, and Mingli Song. 2024. Spatiotemporal-Augmented Graph Neural Networks for Human Mobility Simulation. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[42] Tonglong Wei, Youfang Lin, Shengnan Guo, Yan Lin, Yiheng Huang, Chenyang Xiang, Yuqing Bai, and Huaiyu Wan. 2024. Diff-RNTraj: A Structure-aware Diffusion Model for Road Network-constrained Trajectory Generation. *IEEE Transactions on Knowledge and Data Engineering* (2024), 1–15. https://doi.org/10.1109/TKDE.2024.3460051

[43] Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. 2023. AR-Diffusion: Auto-Regressive Diffusion Model for Text Generation. *Advances in Neural Information Processing Systems* 36 (2023), 39957–39974.

[44] Liu Xi, Chen Hanzhou, and Andris Clio. 2018. trajGANs: using generative adversarial networks for geo-privacy protection of trajectory data. *Vision paper* (2018).

[45] Tianqi Xia, Xuan Song, Zipei Fan, Hiroshi Kanasugi, QuanJun Chen, Renhe Jiang, and Ryosuke Shibasaki. 2018. DeepRailway: A Deep Learning System for Forecasting Railway Traffic. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. 51–56. https://doi.org/10.1109/MIPR.2018.00017

[46] Gang Xiong, Zhishuai Li, Meihua Zhao, Yu Zhang, Qinghai Miao, Yisheng Lv, and Fei-Yue Wang. 2024. TrajSGAN: A Semantic-Guiding Adversarial Network for Urban Trajectory Generation. *IEEE Transactions on Computational Social Systems* 11, 2 (2024), 1733–1743. https://doi.org/10.1109/TCSS.2023.3235923

[47] Yuan Yuan, Jingtao Ding, Chenyang Shao, Depeng Jin, and Yong Li. 2023. Spatio-Temporal Diffusion Point Processes. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3173–3184.

[48] Yuan Yuan, Jingtao Ding, Huandong Wang, Depeng Jin, and Yong Li. 2022. Activity Trajectory Generation via Modeling Spatiotemporal Dynamics. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4752–4762.

[49] Kunpeng Zhang, Xiaoliang Feng, Lan Wu, and Zhengbing He. 2022. Trajectory Prediction for Autonomous Driving Using Spatial-Temporal Graph Attention Transformer. *IEEE Transactions on Intelligent Transportation Systems* 23, 11 (2022), 22343–22353.

[50] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. 2024. Dynamic Diffusion Transformer. *arXiv preprint arXiv:2410.03456* (2024).

[51] Yuanshao Zhu, Yongchao Ye, Yi Liu, and James J. Q. Yu. 2022. Cross-Area Travel Time Uncertainty Estimation From Trajectory Data: A Federated Learning Approach. *IEEE Transactions on Intelligent Transportation Systems* 23, 12 (Dec. 2022), 24966–24978.

[52] Yuanshao Zhu, Yongchao Ye, Shiyao Zhang, Xiangyu Zhao, and James Yu. 2023. 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., 65168–65188.

[53] Yuanshao Zhu, James Jianqiao Yu, Xiangyu Zhao, Qidong Liu, Yongchao Ye, Wei Chen, Zijian Zhang, Xuetao Wei, and Yuxuan Liang. 2024. ControlTraj: Controllable Trajectory Generation with Topology-Constrained Diffusion Model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4676–4687.

A Design Principle of TDFormer Block

The TDFormer block is a novel architectural design that integrates adaptive Trajectory Layer Normalization (adaTLN) with Multi-Head Self-Attention (MHSA) to enhance the modeling of trajectory-specific conditions and spatial-temporal dependencies. By combining these two components, TDFormer enables precise trajectory generation while maintaining high fidelity and adaptability to various motion constraints. The adaTLN mechanism provides dynamic normalization conditioned on trajectory-specific contexts, ensuring that the model can generalize across different trajectory patterns. Meanwhile, MHSA captures intricate dependencies within the trajectory sequences, enabling the model to recognize both short-term and long-term relationships. This synergistic integration ensures a robust and structured representation of trajectory data, facilitating more accurate and controlled generation.

The TDFormer block is primarily composed of two key modules: MHSA and adaTLN, each serving a distinct role in the trajectory generation process. MHSA is responsible for extracting spatial-temporal dependencies by computing attention scores across multiple heads, effectively capturing both local and global trajectory correlations. This mechanism allows different attention heads to focus on different aspects of the trajectory data, ensuring a comprehensive feature representation. On the other hand, adaTLN serves as a condition-driven normalization strategy, allowing the model to adapt dynamically to different trajectory conditions. By introducing normalization parameters that vary according to trajectory-specific context, adaTLN provides greater flexibility in controlling trajectory generation.

MHSA plays a pivotal role in encoding trajectory sequences by facilitating the aggregation of multi-scale information. Traditional sequence models struggle to balance short-term motion variations with long-term dependencies, but MHSA overcomes this challenge by leveraging multiple attention heads that process information at varying granularities. This ability to model complex relationships within trajectory sequences makes MHSA particularly well-suited for generative tasks that require high fidelity and contextual consistency. By integrating MHSA within the TDFormer block, the model can dynamically adjust its attention distribution, ensuring that trajectory features are effectively captured and utilized for generation.

The adaTLN mechanism further enhances the adaptability of TDFormer by incorporating condition-specific normalization adjustments. Unlike conventional layer normalization techniques, which apply fixed normalization parameters, adaTLN dynamically adjusts the scale ($\boldsymbol{\gamma}$) and shift ($\boldsymbol{\beta}$) parameters based on external trajectory conditions. This mechanism is implemented using a scale-and-shift module, which processes embeddings from the time step t and the condition c to compute the appropriate transformation parameters. By aligning the normalization process with trajectory conditions, adaTLN ensures that the generated trajectories adhere to specified constraints while preserving natural motion patterns.

To further enhance model stability and adaptability, TDFormer incorporates dimension-wise scaling parameters ($\boldsymbol{\alpha}$) just before each residual connection. These scaling parameters play a crucial role in modulating the contribution of different components during the training process. By initializing $\boldsymbol{\alpha}$ to zero at the beginning of training, we suppress the premature influence of newly introduced components, allowing the model to gradually adapt to conditional variations. As training progresses, $\boldsymbol{\alpha}$ is optimized to control the magnitude of conditional modifications, ensuring progressive adaptation while maintaining training stability. This strategy not only prevents sudden shifts in trajectory representations but also improves the convergence properties of the model.

The flow of these parameters within the TDFormer block follows a structured process, where condition variables are first embedded and passed through a multi-layer perceptron (MLP) to generate the necessary normalization parameters. The computed scale-and-shift parameters are then used to adjust the trajectory representations, ensuring that the generated trajectories are contextually aligned with the given conditions.

By leveraging the combination of MHSA for dependency modeling and adaTLN for conditional normalization, TDFormer ensures that generated trajectories adhere to realistic spatial-temporal constraints while preserving fine-grained controllability. Furthermore, the integration of adaptive scaling mechanisms provides additional training stability, making TDFormer well-suited for large-scale trajectory prediction and synthesis tasks. Through this structured and iterative approach, the TDFormer block significantly enhances the fidelity, interpretability, and robustness of trajectory generation models.

B Details of Trajectory Conditions

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. These variables provide structured supervision, enabling models to align generated trajectories with user-specified conditions while maintaining high fidelity. By incorporating these guidance variables, the generative model can learn not only the underlying motion patterns but also the contextual dependencies essential for realistic and diverse trajectory synthesis.

Guidance variables are particularly crucial for training, as they serve as external control signals that enhance the controllability, interpretability, and adaptability of the generated outputs. Without these well-defined constraints, the model may produce trajectories that are plausible but lack contextual relevance, leading to limited practical applicability. Therefore, it is essential to carefully design condition variables that effectively guide both the training and inference processes of the generative model. These condition variables help steer the model toward generating realistic, context-aware trajectories that adhere to specific constraints, ensuring high-quality and meaningful outputs.

To achieve this, we introduce two forms of condition variables: a simplified version that includes a comprehensive set of trajectory descriptors, and a real-world version that captures only the most critical factors for practical applications.

$$\boldsymbol{c}_{\text{less}} = \{dt, td, tt, tl, oID, dID\}, \tag{18}$$

$$\boldsymbol{c} = \{dt, oID, dID\}. \tag{19}$$

Here, $\boldsymbol{c}_{\text{less}}$ represents the full set of condition variables used for trajectory generation. In contrast, the real-world condition variables $\boldsymbol{c}$ represent a minimal yet practical subset, including only departure time (*dt*) and the origin-destination pair (*oID*, *dID*). This reduced set reflects the most commonly available and actionable features in real-world applications, ensuring that the generative model remains efficient and adaptable while still producing trajectories that align with observed movement patterns.

By distinguishing between these two sets of condition variables, we provide a flexible approach to trajectory generation, allowing models to operate in both highly controlled and more realistic, data-driven environments. The inclusion of spatial-temporal constraints improves the quality, realism, and usability of the generated trajectories, making them valuable for applications such as traffic simulation, human mobility analysis, and autonomous navigation.

C FACT Implementation Details

In FACT, we perform extensive experiments to determine the optimal parameter and training settings. Table 6 presents the selected hyperparameter values along with their reference ranges, derived from both empirical results in our study and general practices in trajectory generation models.

Table 6: Hyperparameters for FACT.

Parameter	Setting value	Refer range
Diffusion Steps	500	$300 \sim 500$
Skip steps	5	$1 \sim 10$
Hidden dimension	128	≥ 64
β (linear schedule)	$0.0001 \sim 0.05$	–
Batch size	512	≥ 64
Transformer block	18	$10 \sim 28$

Besides, based on varying model depths and sizes, we define three model variants—FACT-S, FACT-M, and FACT-L—to facilitate the discussion on model scaling, as illustrated in Table 7.

Table 7: Details of FACT models. We follow DiT [27] model configurations for the Small (S), Medium (M), and Large (L) variants.

Model	Layers N	Hidden size d	Heads	GFlops
FACT-S	10	64	8	0.2
FACT-M	18	128	16	1.4
FACT-L	28	128	16	2.2

D Experiments and Setup

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

D.1 Datasets

We verify the performance of FACT and baseline models across three distinct urban datasets: Chengdu, Xi'an, and Porto. presents the statistical overview of these datasets is shown in Table 8.

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.

Table 8: 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

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.

D.2 Baselines

- **CVAE** [8]: 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** [25]: 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** [20]: 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** [52]: 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** [42]: 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** [53]: A pretrained Masked AutoEncoder (MAE) encodes the topology information of road segments traversed by trajectory data to improve fidelity. The topology information is

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

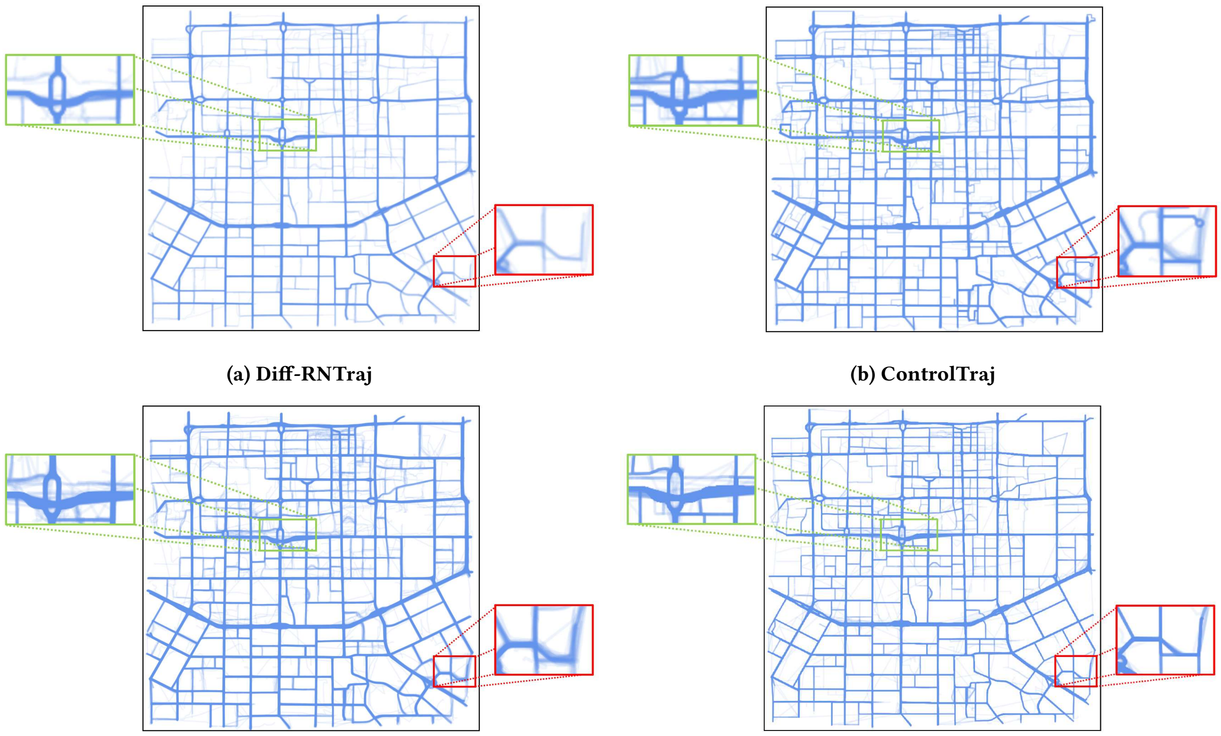
(a) Diff-RNTraj **(b) ControlTraj**
(c) FACT **(d) Real**

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

used during both training and inference to guide the generation process.

D.3 Evaluation Metrics

To quantify the similarity between generated and actual trajectories, a rigorous evaluation of their resemblance is essential. We employ Jensen-Shannon divergence (JSD) as a measure of trajectory quality. JSD effectively compares the distributions of real and synthetic trajectories, where a lower JSD value indicates a closer alignment with the statistical properties of the original data. For evaluation, we apply grid-based statistics by dividing the city into 16x16 grids for distribution counting. The metrics are illustrated as below.

- **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.

D.4 Visualization

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.

D.5 Efficiency Study

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 256. For a more

Table 9: 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**	326	<u>270</u>	644
# Params (million)	27.1774	8.1793	**5.7988**	<u>8.0518</u>
Density (↓)	0.0371	0.0031	**0.0006**	<u>0.0008</u>
Length (↓)	0.1078	0.0097	**0.0044**	<u>0.0049</u>
Pattern (↑)	0.6094	0.8547	**0.8869**	<u>0.8854</u>

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

comprehensive comparison, we present both the time efficiency and the data generation performance on the Chengdu dataset, partially adopted from Table 1.

Based on the results in Table 9, 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.