A DiffTraj Details & Hyperparameters

In this section, we cover the specific details of DiffTraj, including the DiffTraj framework, the Traj-UNet structure, and the implementation details.

A.1 Architecture

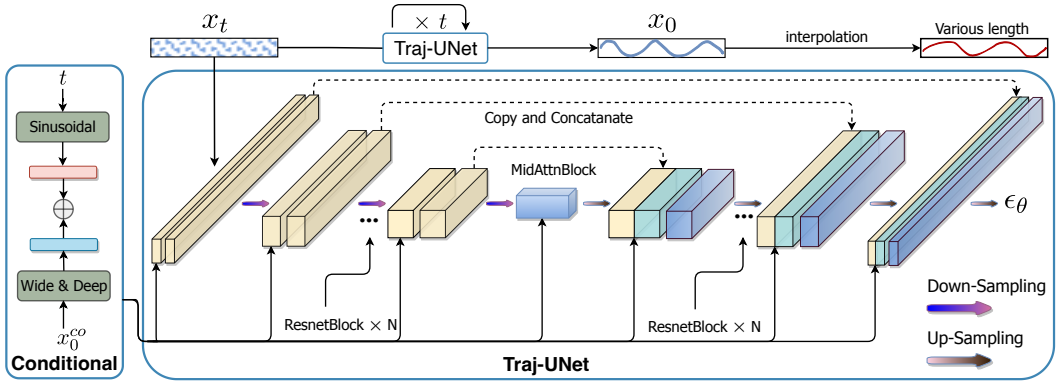


Figure 4: The network architecture used by DiffTraj in modeling $\boldsymbol{\epsilon}_\theta\left(\boldsymbol{x}_t^{\mathrm{s}}, t \mid \boldsymbol{x}_0^{\mathrm{co}}\right)$ is divided into two modules, down-sampling and up-sampling, each containing multiple Resnet blocks.

As illustrated in Fig. 4, DiffTraj is divided into two modules, i.e., down-sampling and up-sampling, and conditional module. Each down-sampling and up-sampling module consists of multiple stacked Resnet blocks. Between the two of them, a transitional module based on the attention mechanism is integrated. To better learn the noise of each time step and guide the generation, DiffTraj integrates a conditional module to embed the time step and external traffic information, later fed to each block. Since the CNN structure can only accept data of fixed shape, we first sample each trajectory as a tensor of $[2,\ \mathrm{length}]$. Specifically, if the trajectory is below the set length, it is added using linear interpolation, and if it is greater than that, the redundant portion is removed using linear interpolation. Thus, the model will generate fixed-length trajectories, which will then be tailored to the desired length by the conditional information.

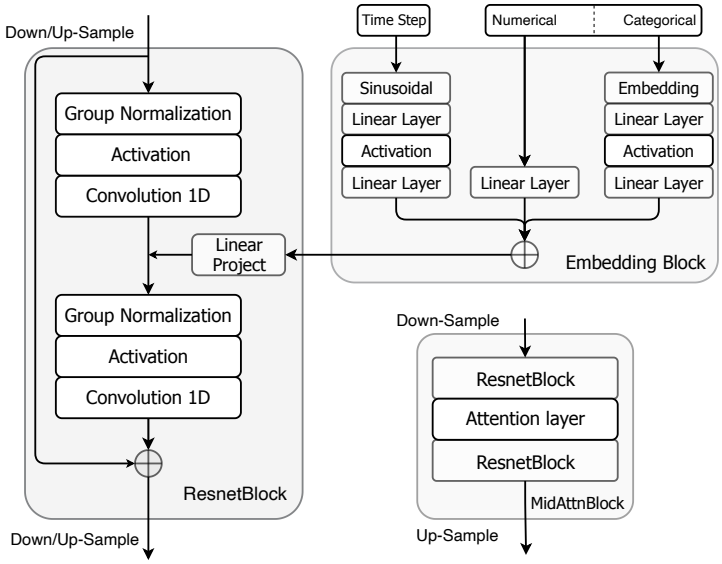


Figure 5: The main components of Traj-UNet including Resnet block, Embedding block, and middle attention block.

The main blocks of Traj-UNet are presented in Fig. 5, i.e., Resnet block, Embedding block, and middle attention block. Among them, each sampling block (down-sampling and up-sampling) consists of multiple Resnet blocks, each containing a series of group normalization, nonlinear activation, and 1D-CNN layers. Then, Traj-UNet applies up-sampling or down-sampling to the output, where down-sampling uses max pooling and up-sampling uses interpolation. After this, Traj-UNet integrates a middle attention block, which consists of two Resnet blocks and an attention layer. Note that there are no additional down-/up-sampling operations in the Resnet block. Finally, we integrate a conditional embedding block to learn the diffusion time step and conditional information. For the diffusion step, we employ Sinusoidal embedding to represent each t as a 128-dimensional vector, and then apply two shared-parameter fully connected layers. For conditional information, such as distance, speed, departure time, travel time, trajectory length, and starting and ending locations, we use the Wide & Deep module for embedding. After getting the diffusion step embedding and conditional embedding, we sum them up and add them to each Resnet block.

A.2 Implementation Details

For the proposed DiffTraj framework, we summarize the adopted hyperparameter settings in Table 3.

Table 3: Hyperparameters setting for DiffTraj.

Hyperparameter	Setting value
Diffusion Steps	500
Skip steps	5
Guidance scale	3
β (linear)	$0.0001 \sim 0.05$
Batch size	1024
Sampling blocks	4
Resnet blocks	2
Input Length	200

The training and sampling phase of the proposed framework is summarized in Algorithm 1 and Algorithm 2, respectively. The detailed implementation code is attached in Supplementary.

Algorithm 1 Diffusion Training Phase

for $i = 1, 2, \ldots,$ **do**
  Sample $\boldsymbol{x}_0 \sim q(\boldsymbol{x})$,
  $t \sim \text{Uniform}\,\{1, \ldots, T\}$
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  $\mathcal{L} = \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t} \boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon},\ t \right) \right\|^2$
  $\theta = \theta - \eta \nabla_\theta \mathcal{L}$
end for

Algorithm 2 Diffusion Sampling Phase

Sample $\boldsymbol{x}_T^{\mathrm{s}} \sim \mathcal{N}(0, \mathbf{I})$
for $t = T, T - S, \ldots, 1$ **do**
  Compute $\boldsymbol{\mu}_\theta \left(\boldsymbol{x}_t^{\mathrm{s}}, t \mid \boldsymbol{x}_0^{\mathrm{co}} \right)$ according to Eq. (5)
  Compute $p_\theta \left(\boldsymbol{x}_{t-1}^{\mathrm{s}} \mid \boldsymbol{x}_t^{\mathrm{s}}, \boldsymbol{x}_0^{\mathrm{co}} \right)$ according to Eq. (4)
end for
return $\boldsymbol{x}_0$

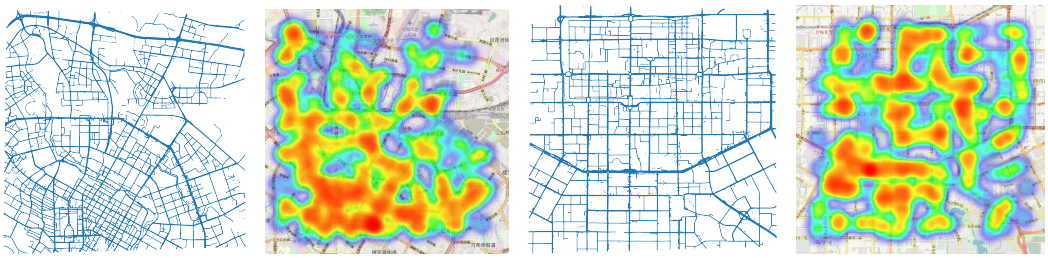
B Details of the Experimental Setup

B.1 Dataset

We evaluate the performance of DiffTraj and all baselines methods on two datasets with different cities, **Chengdu** and **Xi'an**[1]. Both datasets are collected from cab trajectory data starting from November 1, 2016, to November 30, 2016. Table 4 summarizes the statistical information of these two datasets, and Fig. 6 shows the trajectory distribution and heat map of these two datasets, where the deeper color indicates the more concentrated trajectory in the region. For all datasets, we remove all trajectories with lengths less than 120 and sample them to a set fixed length.

Table 4: Statistics of Two Real-world Trajectory Datasets.

Dataset	Trajectory Number	Average Time	Average Distance
Chengdu	3 493 918	11.42 min	7.42 km
Xian	2 180 348	12.58 min	5.73 km



(a) Chengdu Trajectory (b) Chengdu Heatmap (c) Xi'an Trajectory (d) Xi'an Heatmap

Figure 6: Origin trajectory distribution of two cities.

B.2 Evaluation Metrics

As trajectory generation aims to generate trajectories that can replace real-world activities and further benefit downstream tasks, we need to evaluate the "similarity" between the generated trajectories and real ones. In this work, we follow the common practice in previous studies [8] and measure the quality of the generated ones by Jenson-Shannon divergence (JSD). JSD compares the distribution of the real and generated trajectories, and a lower JSD indicates a better match with the original statistical features. Suppose that the original data has a probability distribution P and the generated data has a probability distribution G, the JSD is calculated as follows:

$$\mathrm{JSD}(P, G) = \frac{1}{2}\mathbb{E}_P\left[\log\frac{P}{P+G}\right] + \frac{1}{2}\mathbb{E}_G\left[\log\frac{G}{G+P}\right]. \tag{8}$$

For the evaluation, we divided each city into 16×16 size grids and recorded the corresponding values for each grid. Based on this, we adopt the following metrics to evaluate the quality of the generated trajectories from four perspectives:

- **Density error:** This a global level metric that used to evaluate the geo-distribution between the entire generated trajectory $\mathcal{D}\left(\mathcal{T}_{\text{gen}}\right)$ and the real trajectory $\mathcal{D}(\mathcal{T})$.

$$\text{Density Error} = \mathrm{JSD}\left(\mathcal{D}(\mathcal{T}), \mathcal{D}\left(\mathcal{T}_{\text{gen}}\right)\right), \tag{9}$$

  where $\mathcal{D}(\cdot)$ denotes the grid density distribution in a given trajectory set, and $\mathrm{JSD}(\cdot)$ represents the Jenson-Shannon divergence between two distributions.

- **Trip error:** This a trajectory level metric that measures the correlation between the starting and ending points of a travel trajectory. Specifically, we calculate the probability distribution of the start/end points in the original and generated trajectories and use JSD to measure the difference between them.

[1]These datasets can be downloaded at https://outreach.didichuxing.com/

- **Length error:** This a trajectory level metric to evaluate the distribution of travel distances. It can be obtained by calculating the Euclidean distance between consecutive points.
- **Pattern score:** This is a semantic level metric defined as the top-n grids that occur most frequently in the trajectory. We define P and P_{gen} to denote the original and generated pattern sets, respectively, and compute the following metrics:

$$\text{Pattern score} = 2 \times \frac{\text{Precision}\left(P, P_{\text{gen}}\right) \times \text{Recall}\left(P, P_{\text{gen}}\right)}{\text{Precision}\left(P, P_{\text{gen}}\right) + \text{Recall}\left(P, P_{\text{gen}}\right)} \tag{10}$$

B.3 Baselines

In this section, we introduce the implementation of baseline methods.

- **Random Perturbation (RP):** We generate $-0.01 \sim 0.01$ random noise to perturb the original trajectory. This degree of noise ensures that the maximum distance between the perturbed points and the original does not exceed $2\,\text{km}$
- **Gaussian Perturbation (GP):** We generate Gaussian noise perturbed original trajectories with mean 0 and variance 0.01.
- **Variational AutoEncoder (VAE) [15, 30]:** In this work, trajectories are first embedded as a hidden distribution through two consecutive convolutional layers and a linear layer. Then, we generate the trajectories by a decoder consisting of a linear layer and two deconvolutional layers. The size of convolution kernels in convolutional and deconvolutional layers is set to 4 to ensure that input and output trajectories have the same size.
- **TrajGAN [9]:** The trajectory is first combined with random noise and then passes through a generator consisting of two linear layers and two convolution layers. Subsequently, a convolutional layer and a linear layer are adopted as the discriminator. The generator and the discriminator are trained in an alternating manner.
- **Diffwave [13]:** Diffwave is a Wavenet structure model designed for sequence synthesis, which employs extensive dilated convolution. Here, we use 16 residual connected blocks, each consisting of a bi-directional dilated convolution. Then they are summed using sigmoid and tanh activation, respectively, and fed into the 1D CNN.
- **Diff-scatter:** We randomly sample GPS scatter points from trajectories and generate scatter points using a 4-layer MLP (neutrons: $\{128, 256, 256, 128\}$) and the diffusion model.
- **Diff-wo/UNet:** This model uses only two Resnet blocks combined with a single attention layer between them. Compared with DiffTraj, this model does not have a UNet-type structure, which can be used to evaluate the necessity of the UNet structure.
- **Diff-LSTM:** This model has the same UNet-type structure and number of Resnet blocks as DiffTraj, and the difference is that Diff-LSTM replaces the CNN with LSTM in Resnet block.
- **DiffTraj-wo/Con:** DiffTraj-wo/Con represents that the Wide & Deep conditional information embedding module is removed. The rest is the same as DiffTraj.

C Additional Experiments

C.1 Data Utility Setup

In this paper, we use inflow/outflow traffic forecasting to test the utility of the generated data. Inflow/outflow traffic forecasting is a critical task in urban traffic management that involves predicting the volume of vehicles entering (inflow) or leaving (outflow) a specific region within a certain period of time. In this experiment, we divided a city into 16×16 grids, where each grid represents a specific region. The traffic volume entering (inflow) or leaving (outflow) each region within a certain period is predicted. The primary goal of this experiment is to train various prediction models using both original and generated trajectory data, comparing their prediction performance. This evaluation provides an important perspective on the real-world applicability of the data generated by DiffTraj, assessing not just the fidelity of the generated trajectories, but also their utility in downstream tasks. In the experimental setup, we train the prediction models using the generated data and the original data separately, and then test their prediction performance on real data. Advanced neural network models, such as AGCRN, Graph WaveNet, DCRNN, and MTGNN, have been employed to handle this task due to their ability to capture complex spatial-temporal dependencies in multivariate time series data. All the above models and code in this section are followed the publicly available code[2] provided in the literature [12].

- **AGCRN (Adaptive Graph Convolutional Recurrent Network):** AGCRN is a sophisticated model for spatial-temporal forecasting, which leverages both spatial and temporal features of data. It uses graph convolution to capture spatial dependencies and RNNs to model temporal dynamics, making it capable of handling complex spatial-temporal sequences.
- **GWNet (Graph WaveNet):** GWNet is designed for high-dimensional, structured sequence data. It incorporates a Graph Convolution Network (GCN) to model spatial correlations and a WaveNet-like architecture to model temporal dependencies. The combination allows for capturing both the spatial and temporal complexities present in high-dimensional data.
- **DCRNN (Diffusion Convolutional Recurrent Neural Network):** DCRNN is a deep learning model designed for traffic forecasting, which handles the spatial and temporal dependencies in traffic flow data. It uses a diffusion convolution operation to model spatial dependencies and a sequence-to-sequence architecture with scheduled sampling and residual connections to model temporal dependencies.
- **MTGNN (Multivariate Time-series Graph Neural Network):** MTGNN is a model that captures complex spatial-temporal relationships in multivariate time series data. The model leverages a graph neural network to model spatial dependencies and an auto-regressive process to capture temporal dependencies. It also uses a gating mechanism to adaptively select the relevant spatial-temporal components, thus improving the forecasting performance.

For accuracy comparison, we use the mean square error (MSE), root mean square error (RMSE), and mean absolute error (MAE) as metrics to assess the prediction accuracy of all methods. These three metrics are defined as follows:

$$\mathrm{MSE}(\mathbf{X},\ \hat{\mathbf{X}}) = \frac{1}{N}\sum_{i}^{N}\left(X^{(i)} - \hat{X}^{(i)}\right)^2, \tag{11}$$

$$\mathrm{RMSE}(\mathbf{X},\ \hat{\mathbf{X}}) = \sqrt{\frac{1}{N}\sum_{i}^{N}\left(X^{(i)} - \hat{X}^{(i)}\right)^2}, \tag{12}$$

$$\mathrm{MAE}(\mathbf{X},\ \hat{\mathbf{X}}) = \frac{1}{N}\sum_{i}^{N}\left|X^{(i)} - \hat{X}^{(i)}\right|, \tag{13}$$

where $X^{(i)}$ and $\hat{X}^{(i)}$ are the ground truth and predicted inflow/outflow value at time i, respectively.

[2] https://github.com/deepkashiwa20/DL-Traff-Graph

C.2 Conditional Generation

As we described in Sec. 4.2 and Sec. 4.3, the DiffTraj framework takes into account various external factors that influence real-world trajectories, such as road network structure and departure time. These factors are used to guide the generation process, ensuring the synthetic trajectories mimic real-world patterns and behaviors. The model employs a Wide & Deep network to effectively embed this conditional information, enhancing the capabilities of the Traj-UNet model.

To evaluate the conditional generation capability of DiffTraj, we investigate the case of generated trajectories where the start and end regions of the trajectories were pre-defined. The model was tasked to generate 20 random trajectories adhering to these conditions. The results depicted in Fig. 7 and Fig. 8 (The red and blue boxes indicate the starting and ending regions, respectively), effectively demonstrate proficiency in generating trajectories that align with the specified start and end regions of DiffTraj. This is observed consistently across both cities under study, reinforcing the model's ability to accurately incorporate and adhere to conditional information. This robust capability underscores the versatility of DiffTraj in generating meaningful trajectories under specific conditions, and its applicability in real-world scenarios where such conditions often exist.

C.3 Ensuring Generation Diversity

In addition, DiffTraj is designed to generate high-quality trajectories and ensure a level of diversity that prevents overly deterministic behavior patterns, thereby upholding intended privacy protections. By integrating a classifier-free diffusion guidance method, DiffTraj can strike a calculated balance between sample quality and diversity. To validate the capacity for generating diverse trajectories of DiffTraj, we devised an experiment that manipulates the guiding parameter, ω. This experiment aimed to examine the quality-diversity balance in trajectories generated by DiffTraj, and how this equilibrium responds to variations in ω. In this experimental setup, we studied the trajectories yielded under different ω settings (specifically $\omega \in 0.1, 1, 10$) while keeping the conditional information the same.

The results, as illustrated in Fig. 7 and Fig. 8, reveal an unambiguous link between an increase in ω and a rise in trajectory diversity. This finding affirms that DiffTraj adeptly manages the balance between diversity and quality. As ω increases, the model demonstrates a tendency to spawn more varied trajectories. This is because a higher ω prompts the model to place more emphasis on unconditional noise prediction and reduce the sway of the conditional information. Thus, the model grows more proficient at creating diverse trajectories, albeit potentially compromising some quality. This greater diversity is a consequence of the model having fewer constraints from specific conditions, providing more latitude to explore a wider range of possible trajectories. This experiment underscores the flexibility and control inherent in DiffTraj in balancing trajectory quality and diversity, vital characteristics for generating realistic and diverse trajectories. Therefore, ω serves as a control knob for modulating the trade-off between trajectory quality and diversity, providing a powerful tool for users to align the generated trajectories with specific application requirements.

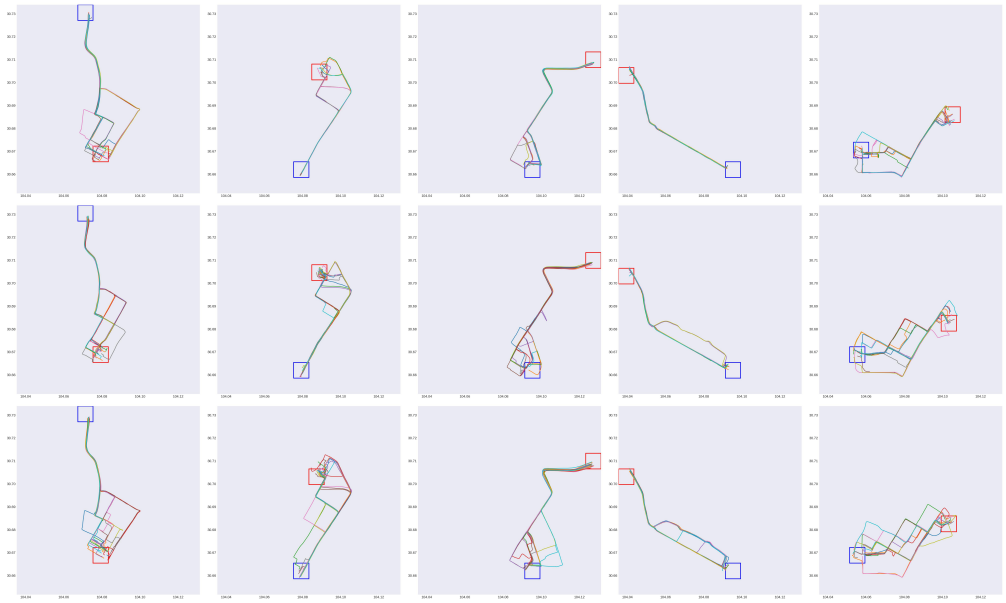


Figure 7: Conditional trajectory generation on Chengdu. The guidance scales ω of the first, second and third rows are $0.1, 1, 10$, respectively. The rectangular box indicates the area of the assigned start and end points.

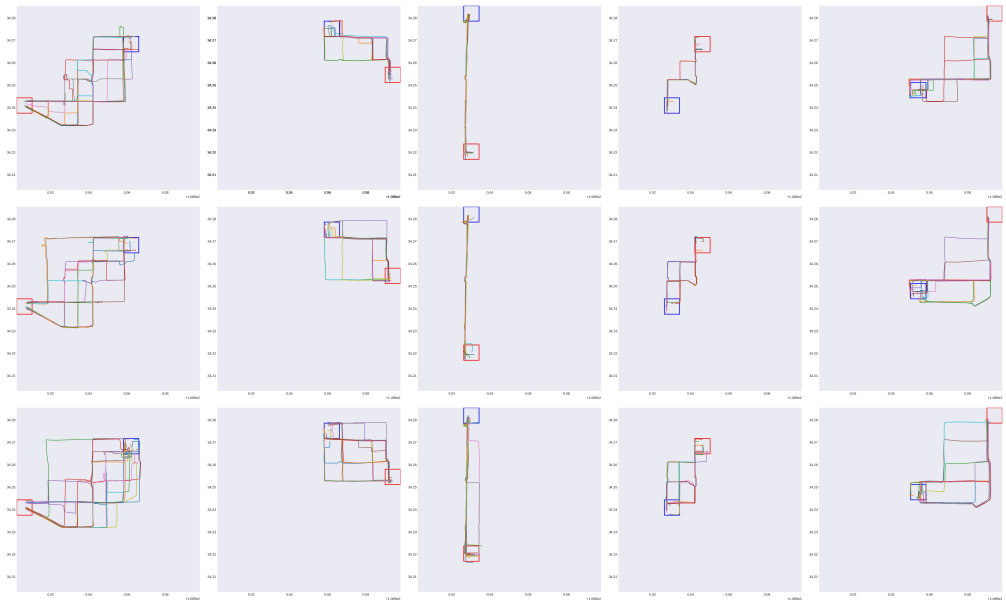
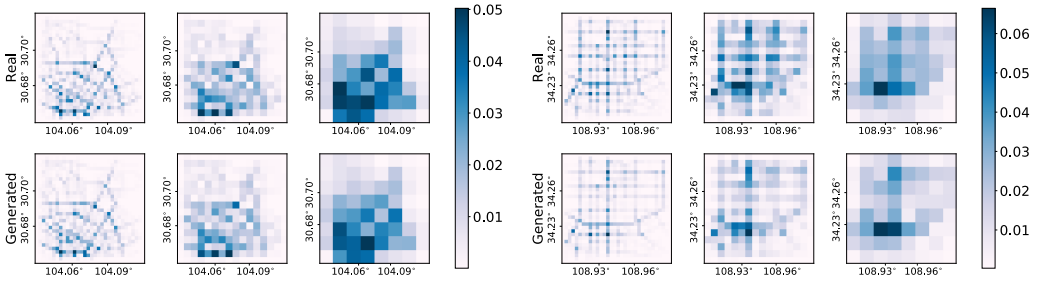


Figure 8: Conditional trajectory generation on Xi'an. The guidance scales ω of the first, second and third rows are $0.1, 1, 10$, respectively. The rectangular box indicates the area of the assigned start and end regions.

D Visualization Results

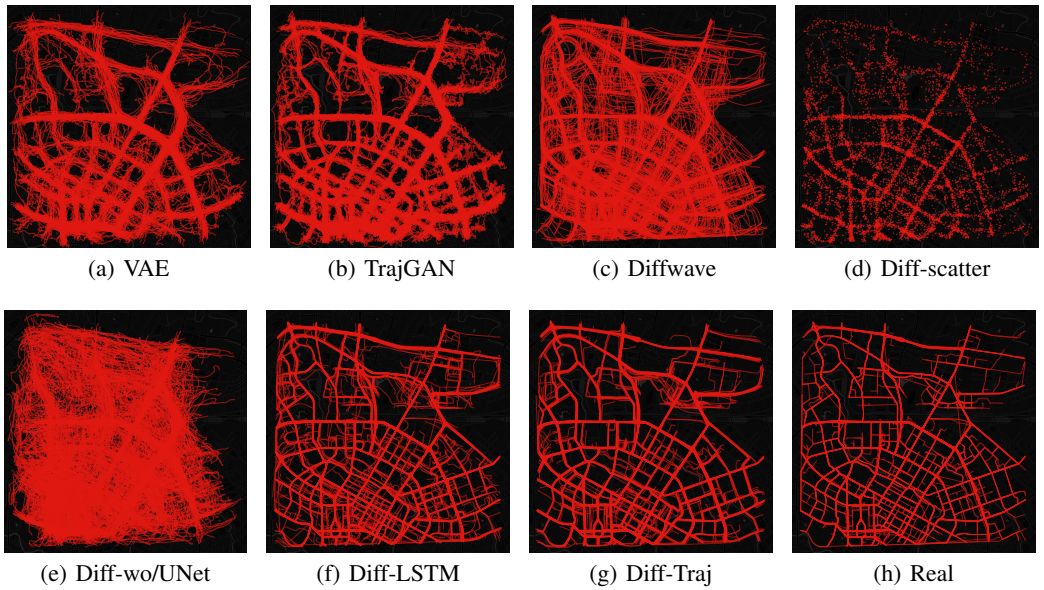
We append a series of experimental results in this section due to space constraints. As shown in Fig. 9, we visualize the heat map of the trajectory distribution with multiple resolutions. Specifically, we divide the whole city into 32×32, 16×16, and 8×8 grids, and then count the distribution of trajectories in each grid. The comparison clearly indicates that the distributions are highly consistent from all resolutions. The visualized results also verify the effectiveness of metrics in Table 1, revealing that the proposed model can generate high-quality trajectories with remarkable accuracy and retain the original distribution.



(a) Chengdu. (b) Xi'an.

Figure 9: Comparison of the real and generated trajectory distributions with different resolutions. The city is divided into different size grids (32×32, 16×16, and 8×8 grids).

In addition, we also show the geographic results of the trajectories generated by different generation methods for two cities, Chengdu and Xi'an. The visualization results are concluded in Fig. 10 and Fig. 11.

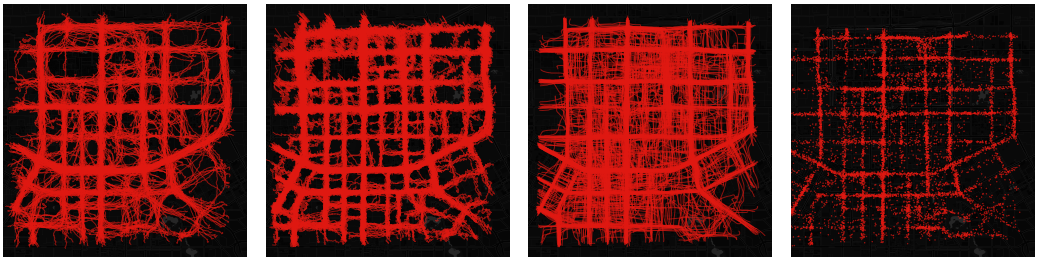


(a) VAE (b) TrajGAN (c) Diffwave (d) Diff-scatter

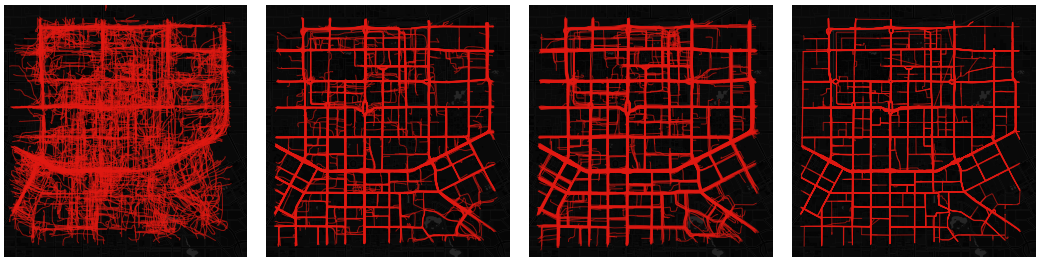
(e) Diff-wo/UNet (f) Diff-LSTM (g) Diff-Traj (h) Real

Figure 10: Geographic visualization of generated trajectory in Chengdu.

The rest visualize the forward trajectory addition noise process and reverse trajectory denoising process of Diff-Traj.



(a) VAE (b) TrajGAN (c) Diffwave (d) Diff-scatter



(e) Diff-wo/UNet (f) Diff-LSTM (g) Diff-Traj (h) Real

Figure 11: Geographic visualization of generated trajectory in Xi'an.

Figure 12: Forward trajectory noising process (Chengdu).

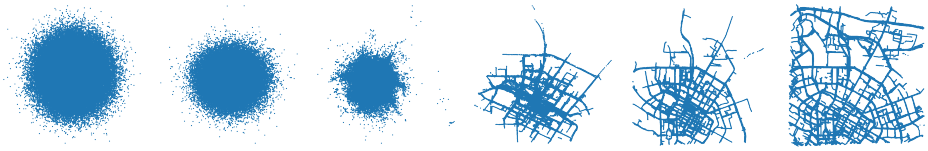


Figure 13: Reverse trajectory denoising process (Chengdu).

Figure 14: Forward trajectory noising process (Xi'an).

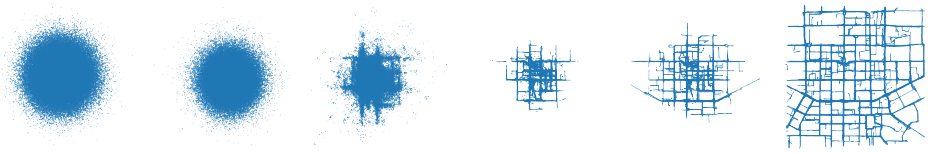


Figure 15: Reverse trajectory denoising process (Xi'an).