Supplementary material

Anonymous submission

Experimental Environment

Our method is implemented in Python and trained using an Nvidia A40 GPU. All experiments are run on the Ubuntu 20.04 operating system with an Intel Xeon Gold 6330 CPU. Python third-party dependency file: `codedata/code/requirements.txt`

Datasets

In the article, we conducted a comprehensive evaluation of our algorithm on two real-world datasets, which are from Shenzhen and Porto, respectively. Below is a brief description of the two datasets:

Shenzhen. (Zhang et al. 2015) released this dataset containing approximately 510k dense trajectories generated by 14k taxi cabs in Shenzhen, China, which can be downloaded at https://people.cs.rutgers.edu/˜dz220/Data.html.

Porto. This dataset describes trajectories performed by 442 taxis running in the city of Porto, Portugal 1 (Moreira-Matias et al. 2013) . Each taxi reports its location every 15s. This dataset is used for the Trajectory Prediction Challenge@ ECML/PKDD 2015.

Due to space constraints and to facilitate easier replication of our work, we have included representative subsets of the two datasets in the codedata. Data path: `/root/codedata/datasets/raw_data under_noise`

Experiment Code

Preprocessing

As described in Figure 3 of the article in Methodology - Vectorization and Linear Decoder, we convert the raw data into corresponding vectorized representations.

Data path:

`/codedata/datasets/Vectorized_data`

Input: Raw data e.g.

`tra_datasetfutian5_5_3892_p0.npy`

Output: Vectorized file e.g.

`Tensortra_datasetfutian5_5_3892_p0.pth`

Code path:

`/codedata/code/generate_tensor.py`

Example of usage:

```
python generate tensor.py -n futian
```

Model Training

Code path: `codedata/code/main.py`

Example of usage:

```
python main.py -n futian -p 0
```

Model Evaluation

The evaluation results of the proposed method will be directly output during the training process.

Comparison Methods

1) GDP (Shi et al. 2024) is a diffusion-based model for path planning and generation that learns path patterns while incorporating road network constraints, which can be considered the state-of-the-art method in road network trajectory generation. The code comes from the authors' repository.

2) MTNet (Wang et al. 2022) is a deep generative model which employs a knowledge-based meta-learning module to learn generalized distribution patterns from skewed trajectory data. The code comes from the authors' repository.

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

Denoising

Code path: `codedata/code/recover_fix_D.py`

Example of usage:

```
python recover_fix_D.py -n futian -p 0
```

References

Moreira-Matias, L.; Gama, J.; Ferreira, M.; Mendes-Moreira, J.; and Damas, L. 2013. Predicting Taxi–Passenger Demand Using Streaming Data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3): 1393–1402.

Shi, D.; Tong, Y.; Zhou, Z.; Xu, K.; Wang, Z.; and Ye, J. 2024. GRAPH-CONSTRAINED DIFFUSION FOR END-TO-END PATH PLANNING.

Wang, Y.; Li, G.; Li, K.; and Yuan, H. 2022. A Deep Generative Model for Trajectory Modeling and Utilization. *Proceedings of the VLDB Endowment*, 16(4): 973–985.

Zhang, D.; Zhao, J.; Zhang, F.; and He, T. 2015. UrbanCPS: A Cyber-Physical System Based on Multi-Source

Big Infrastructure Data for Heterogeneous Model Integration. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, ICCPS '15, 238–247. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-3455-6.