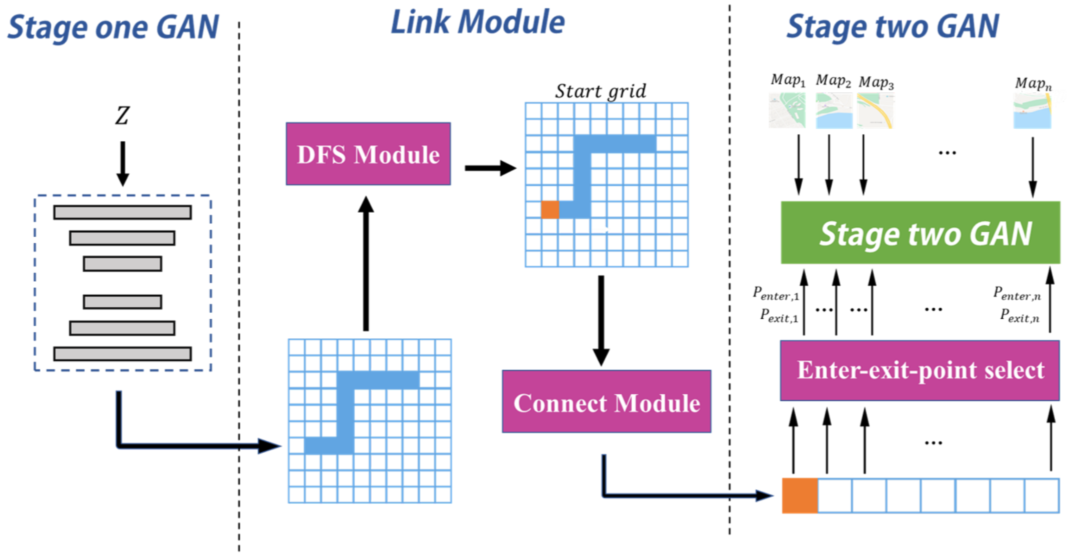
Large Scale GPS Trajectory Generation Using Map based on Two-stage-GAN

Python impletement of paper Large Scale GPS Trajectory Generation Using Map based on Two-stage-GAN

Introduction

we propose a map-based Two-Stage GAN method (TSG) to generate fine-grained and plausible large-scale trajectories. In the first stage, we first transfer GPS points data to discrete grid representation as the input for a modified deep convolutional generative adversarial network to learn the general pattern. In the second stage, inside each grid, we design an effective encoder-decoder network as the generator to extract road information from map image and then embed it into two parallel Long Short-Term Memory networks to generate GPS point sequence.



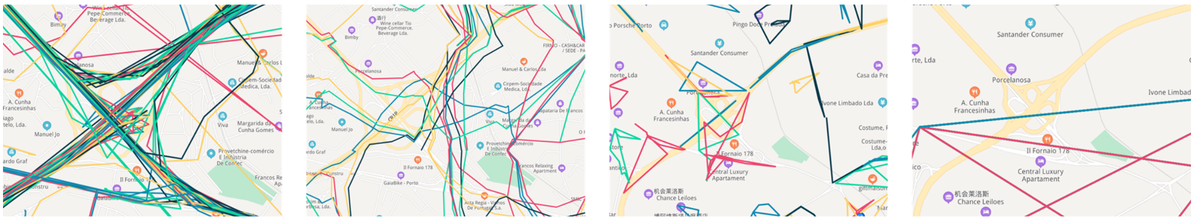
Result

We evaluate the synthetic trajectories in terms of their similarity to real data, i.e., distribution of overall GPS coordinate, distribution of trajectory sequences length, distribution of trajectory distance, top-N visited places and road networks matching accuracy. And we compare our result with the previous benchmark.

JS distance of distribution

Model	$p_o(r)$	$p_s(l)$	$p_d(l)
FTS-IP	0.413	0.182	0.187
LSTM	0.633	**0.058**	0.140
TSG	**0.100**	0.139	**0.136**

Visualization of road network matching



(a) Ground truth (b) Ours (c) FTS (d) LSTM

Train

Dataset

Trajectory data in Porto, [available on Kaggle]()

Prepare the data

1. Transform trajectory data into grids format `pre_process/process_trajectory_data/to_grid.py`
2. Prepare the corresponding map images:
    - go to `pre_process/map_generation/`
    - run `screen_shot.py`
    - run `cut.py`
    - run `merge.py`

First stage GAN

1. go to `First_stage_gan/`.
2. run:

```
python WGANGP.py \
--dataroot ./grid32/ \
--labelroot ./traj_all_0115.txt \
--outf ./output \
--batchSize 64 \
--n_critic 1 \
--netG ./output_IN/netG_epoch_320.pth \
--netD ./output_IN/netD_epoch_320.pth \
--cuda \
--start_iter 320 \
--niter 350
```

Second stage GAN

1. go to `Second_stage_gan`.
2. run `python train.py`.

Generate trajectory data

1. Coarse result generated from First stage GAN

```
cd First_stage_gan
python generate_fake_data.py --large_num 200 --model_path ./output_IN/netG_epoch_260.pth --output_path ../output_generated_coarse
```

2. Final result `TSG/pred.py`

Configurations

- `step_1_output` path to the result of first stage GAN
- `map_dir` path to the map data
- `checkpoint` model result of second stage GAN