250214
Project Meeting: Synthetic Data

AAILAB
Department of Industrial and Systems Engineering
KAIST

Contents

- 전체 (1년치) 데이터를 사용할 수 있도록 데이터 전처리
  - 기존 결과들은 1일치, 7일치 수준에서 모델 학습 진행
  - 현재 받은 데이터 (총 1년치) 를 모두 활용할 수 있도록 데이터 전처리 진행

- Paper review for conditional guidance on discrete diffusion
  - Simple Guidance Mechanisms for Discrete Diffusion Models (ICLR 2025)
    - Discrete diffusion 에서 guidance mechanism 적용 방법

    - Path timestamp condition 적용할 때 활용
    - Continuous diffusion 에서 수행하였던 discriminator 도입 방법 등을 discrete diffusion 에 적용하는 방법에 대한 기반 지식

Data Preprocessing

Original Dataset

timestamp	vehicleid	latitude	longitude	speed
20190701172918	1	36.35223	127.3704	25
20190701172919	1	36.35226	127.3703	21
20190701172920	1	36.3523	127.3702	20

- 기간 전체의 gps point에 할당, path_id정의에 부합 X
- 다른 일자에 같은 vehicle id 존재. (데이터간 중복)

Preprocessed Dataset

path_id	timestamp	latitude	longitude
??	1612224749	36.35223	127.3704
??	1612224750	36.35226	127.3703
??	1612224751	36.3523	127.3702

조건 1 : 특정 duration에 들어올 것.
조건 2 : Loop가 있지 않을 것.

- Goal : 상기 조건을 만족하는 최대한 많은 data point를 남기는 것으로 목표로 함.
- Daejeon-DTG 데이터는 vehicle id로 구분이 되어 있음. 이를 path_id로 적절히 변환해야 함.
  - 이는 위 조건 두개를 위반할 확률이 매우 큼.
    - 조건 1은 refine_gps.py의 convert_single 함수에서 확인 가능.
    - 조건 2는 mapmatching.py의 map_single 함수에서 확인 가능.

Data Preprocessing

파일명 \ 일수	1일치	30일치	1년치
A	3,335,019	3,335,437	3,335,437
G	101,575	101,391	101,391
path	223,539,160	241,545,842	260,968,995
v_path	62,825,356	139,290,932	321,047,320

[일수별 데이터 파일 크기]

* 1일치와 30일치/1년치의 전처리 코드에서 바뀐 부분이 존재함.

- 위와 같이 path, v_path 파일은 일수가 커짐에 따라 데이터 용량도 증가함.
- 그래프 구조에 관한 A, G 파일은 일수가 달라져도 변하지 않는 것으로 확인.

- 하나의 path_id에 할당된 시간 재조정 후 이전 결과와 비교해볼 필요 있을 것으로 보임.

Guidance Mechanism for Discrete Diffusion Models

- Formulation of discrete diffusion models
  - Forward
    - $q(x_t|x_0) = \mathrm{Cat}(x_t; \alpha_t x_0 + (1-\alpha_t)\pi)$ where π is prior; and α_t is a noise schedule, decreasing in t.
  - Reverse (posterior)
    - $q(x_s|x_t, x_0) = \mathrm{Cat}\left(x_s; \frac{[\alpha_{t|s} x_t + (1-\alpha_{t|s})1\pi^T x_t] \odot [\alpha_s x_0 + (1-\alpha_s)\pi]}{\alpha_t x_t^T x_0 + (1-\alpha_t) x_t^T \pi}\right)$ where $\alpha_{t|s} = \alpha_t / \alpha_s$.

- Diffusion guidance
  - Rely on different ways of expressing the score of a distribution conditioned on y.
  - The problem with applying guidance to discrete diffusion is that guidance terms are not differentiable with respect to the discrete representation x_t.

Guidance Mechanism for Discrete Diffusion Models

- Formulation of tempered distribution
  - $p^{\gamma}(x_s|y,x_t) \propto p(y|x_s,x_t)^{\gamma} p_{\theta}(x_s|x_t)$ where γ is temp. parameter and p_{θ} is pre-trained diffusion.
  - ➔ $\log p^{\gamma}(x_s|y,x_t) = \gamma \log p(y|x_s,x_t) + \log p_{\theta}(x_s|x_t) + C$

- Classifier-free guidance
  - Single token: $\log p^{\gamma}(x_s|y,x_t) = \gamma \log p(x_s|y,x_t) + (1-\gamma) \log p(x_s|x_t) + C$
  - Sequences: $\log p^{\gamma}\left(x_s^{(1:L)}\middle|y,x_t^{(1:L)}\right) = \gamma \log p\left(x_s^{(1:L)}\middle|y,x_t^{(1:L)}\right) + (1-\gamma) \log p\left(x_s^{(1:L)}\middle|x_t^{(1:L)}\right) + C$

Independent factorization →

➔ $p^{\gamma}\left(x_s^{(1:L)}\middle|y,x_t^{(1:L)}\right) \propto p\left(x_s^{(1:L)}\middle|y,x_t^{(1:L)}\right)^{\gamma} p\left(x_s^{(1:L)}\middle|x_t^{(1:L)}\right)^{1-\gamma}$

➔ $p_{\theta}^{\gamma}\left(x_s^{(1:L)}\middle|y,x_t^{(1:L)}\right) = \prod_{l=1}^{L} \frac{1}{Z^{(l)}} p_{\theta}\left(x_s^{(l)}\middle|y,x_t^{(1:L)}\right)^{\gamma} p_{\theta}\left(x_s^{(l)}\middle|x_t^{(1:L)}\right)^{1-\gamma}$

Conditional network — Unconditional network

Guidance Mechanism for Discrete Diffusion Models

- Formulation of tempered distribution
  - $p^{\gamma}(x_s|y,x_t) \propto p(y|x_s,x_t)^{\gamma} p_\theta(x_s|x_t)$ where γ is temp. parameter and p_θ is pre-trained diffusion.
  - ➔ $\log p^{\gamma}(x_s|y,x_t) = \gamma \log p(y|x_s,x_t) + \log p_\theta(x_s|x_t) + C$

- Classifier guidance
  - Single token: $p^{\gamma}(x_s|y,x_t) = \frac{p(y|x_s,x_t)^{\gamma} p(x_s|x_t)}{\sum_{x_s'} p(y|x_s',x_t)^{\gamma} p(x_s'|x_t)}$
  - Sequences: $p^{\gamma}\left(x_s^{(1:L)}\middle|y,x_t^{(1:L)}\right) = \prod_{l=1}^{L} p^{\gamma}\left(x_s^{(l)}\middle|y,x_t^{(1:L)}\right)$
    - Where $p^{\gamma}\left(x_s^{(l)}\middle|y,x_t^{(1:L)}\right) = \frac{p\left(y\middle|x_s^{(l)},x_t^{(1:L)}\right)^{\gamma} p\left(x_s^{(l)}\middle|x_t^{(1:L)}\right)}{\sum_{x_s'} p\left(y\middle|x_s',x_t^{(1:L)}\right)^{\gamma} p\left(x_s'\middle|x_t^{(1:L)}\right)} \approx \frac{p_\phi\left(y\middle|\tilde{z}_t^{(1:L)}\right)^{\gamma} p_\theta\left(x_s^{(l)}\middle|x_t^{(1:L)}\right)}{\sum_{x_s'} p_\phi\left(y\middle|\tilde{z}_t^{(1:L)}\right)^{\gamma} p_\theta\left(x_s'\middle|x_t^{(1:L)}\right)}$

Classifier

Unconditional network

$p^{\gamma}\left(y\middle|x_s^{(l)},x_t^{(1:L)}\right) \approx p_\phi\left(y\middle|\tilde{z}_t^{(1:L)}\right)$ where $\tilde{z}_t^{(1:L)} = \left[z_t^{(1:l-1)}; z_s^{(l)}; z_t^{(l+1:L)}\right]$

TODO

- 시간에 따른 교통량 예측을 위하여 diffusion model에 timestamp condition 추가
  - 현재 diffusion model 은 unconditional generation 만 수행
  - 시간을 조건으로 하는 conditional generation 수행

  - 1일치 데이터로 진행하였을 때는 학습이 잘 되지 않았음 ➔ 데이터 문제로 추정
    - 각 시간에 따른 데이터 부족

- Review paper 와 유사한 방식으로 pre-trained & target distribution 의 차이가 있는 상황에서 guidance 방법 고안

APPENDIX

Data Preprocessing

Original Dataset

timestamp	vehicleid	latitude	longitude	speed
20190701172918	1	36.35223	127.3704	25
20190701172919	1	36.35226	127.3703	21
20190701172920	1	36.3523	127.3702	20

- 기간 전체의 gps point에 할당, path_id정의에 부합 X
- 다른 일자에 같은 vehicle id 존재. (데이터간 중복)

Preprocessed Dataset

path_id	timestamp	latitude	longitude
??	1612224749	36.35223	127.3704
??	1612224750	36.35226	127.3703
??	1612224751	36.3523	127.3702

조건 1 : 특정 duration에 들어올 것.
조건 2 : Loop가 있지 않을 것.

- Goal : 상기 조건을 만족하는 최대한 많은 data point를 남기는 것으로 목표로 함.
- Daejeon-DTG 데이터는 vehicle id로 구분이 되어 있음. 이를 path_id로 적절히 변환해야 함.
  - 이는 위 조건 두개를 위반할 확률이 매우 큼.
    - 조건 1은 refine_gps.py의 convert_single 함수에서 확인 가능.
    - 조건 2는 mapmatching.py의 map_single 함수에서 확인 가능.

Data Preprocessing

- **(시도 1)** vehicle_id를 기준으로 150개씩 grouping하여 하나의 path_id라고 고려함.
  - Format : path_id = vehicle_id + group index
  - 전처리 결과
    - 유효 path 수 : 37687
      - 30일 데이터가 38141개. 참고로, Path 수는 돌릴 때마다 달라짐.
    - 소요 시간 : 3802초
  - 실패 지점
    - vehicle_id의 중복이 발생하여, 하나의 path_id에 들어온 다른 데이터셋의 gps 기록은 위 조건 1을 위반하여 버려지게 됨.
    - 주로 gps가 1초 단위로 기록이 되어있어, 150개로 자르면 약 3분 내외의 path_id가 만들어지기에 이 역시 조건 1을 위반함.

Data Preprocessing

- **(시도 2)** vehicle_id 기준 900개씩 grouping 하고 day를 추가하여 하나의 path_id로 고려함.
  - Format : path_id = vehicle_id + group index + day
  - 전처리 결과
    - 유효 path 수 : 8만여개
    - 소요 시간 : 약 16시간

```
['0', 'p1-02019-07-01', '1561969758', '127.370368', '36.352225']
['1', 'p1-02019-07-01', '1561969759', '127.370303', '36.352261']
['2', 'p1-02019-07-01', '1561969760', '127.370248', '36.352296']
['3', 'p1-02019-07-01', '1561969761', '127.370196', '36.352328']
['4', 'p1-02019-07-01', '1561969762', '127.37014', '36.35236']
['5', 'p1-02019-07-01', '1561969763', '127.370075', '36.352396']
['6', 'p1-02019-07-01', '1561969764', '127.370006', '36.352438']
```

  - 실패 지점
    - 1초씩 기록되었다면 900개씩 묶었을 때, 15분정도로 유효한 path가 되어야하나, 몇 개의 data point는 25분을 넘어가는 경우도 보았음. 전반적인 데이터 포인트의 interval을 확인해봐야할 듯.
    - 근본적으로 시간을 늘렸으니, loop가 발생할 위험이 늘어날 수 있음.
  - 방향
    - 몇 개의 gps 포인트를 묶어야 하는지 다시 생각해봐야 할 것.

Data Preprocessing

파일명 \ 일수	1일치	30일치	1년치
A	3,335,019	3,335,437	3,335,437
G	101,575	101,391	101,391
path	223,539,160	241,545,842	260,968,995
v_path	62,825,356	139,290,932	321,047,320

[일수별 데이터 파일 크기]

* 1일치와 30일치/1년치의 전처리 코드에서 바뀐 부분이 존재함.

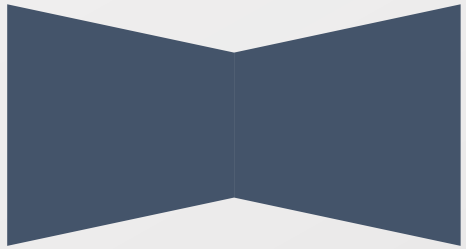
- 위와 같이 path, v_path 파일은 일수가 커짐에 따라 데이터 용량도 증가함.
- 그래프 구조에 관한 A, G 파일은 일수가 달라져도 변하지 않는 것으로 확인.

- 하나의 path_id에 할당된 시간 재조정 후 이전 결과와 비교해볼 필요 있을 것으로 보임.

Time-conditioned U-Net

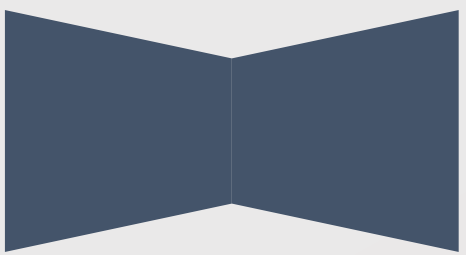
```python
def forward(self, xt_padded, lengths, t):
    # xt_padded: shape b, h, each is a xt label
    # t: shape b
    t = self.time_mlp(t)
    x = self.x_embedding(xt_padded)
    hiddens = []
    for k, down_block in enumerate(self.down_blocks):
        x, h = down_block(x, lengths if k == 0 else None, t)
        hiddens.append(h)

    x = self.mid_block1(x, t)
    x = self.mid_attn(x, None)
    x = self.mid_block2(x, t)

    for up_block in self.up_blocks:
        x = torch.cat((x, hiddens.pop()), dim=-1)
        x, _ = up_block(x, None, t)
    x = self.final_conv(x)
    return x
```

time



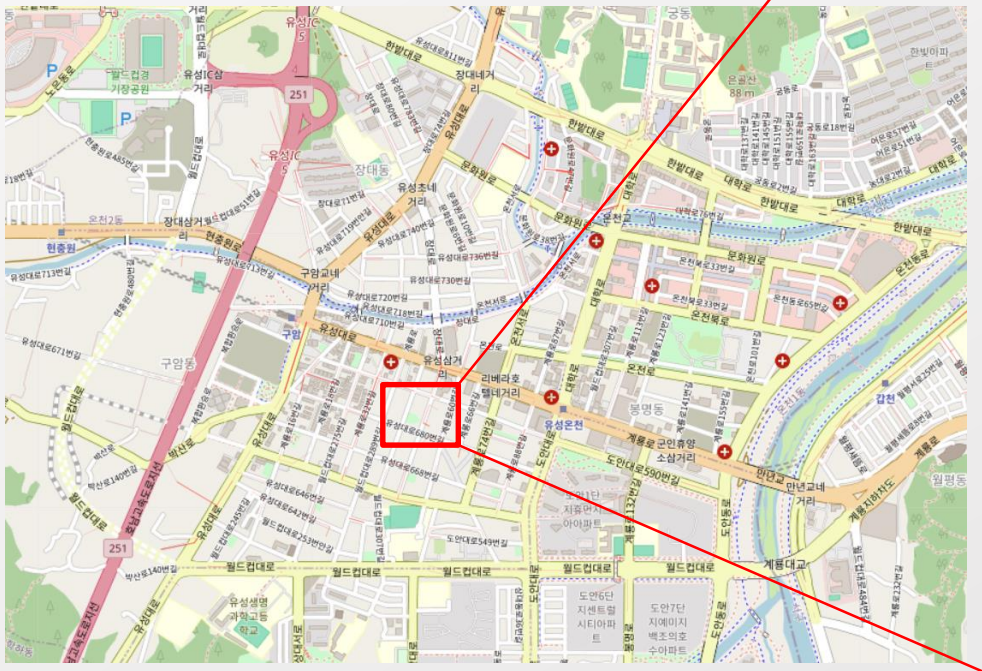
```python
def forward(self, xt_padded, lengths, t, sim_time):
    # xt_padded: shape b, h, each is a xt label
    # t: shape b
    # sim_time: shape b
    t = self.time_mlp(t)
    sim_time = self.sim_time_mlp(sim_time)
    t = t + sim_time
    x = self.x_embedding(xt_padded)
    hiddens = []
    for k, down_block in enumerate(self.down_blocks):
        x, h = down_block(x, lengths if k == 0 else None, t)
        hiddens.append(h)

    x = self.mid_block1(x, t)
    x = self.mid_attn(x, None)
    x = self.mid_block2(x, t)

    for up_block in self.up_blocks:
        x = torch.cat((x, hiddens.pop()), dim=-1)
        x, _ = up_block(x, None, t)
    x = self.final_conv(x)
    return x
```

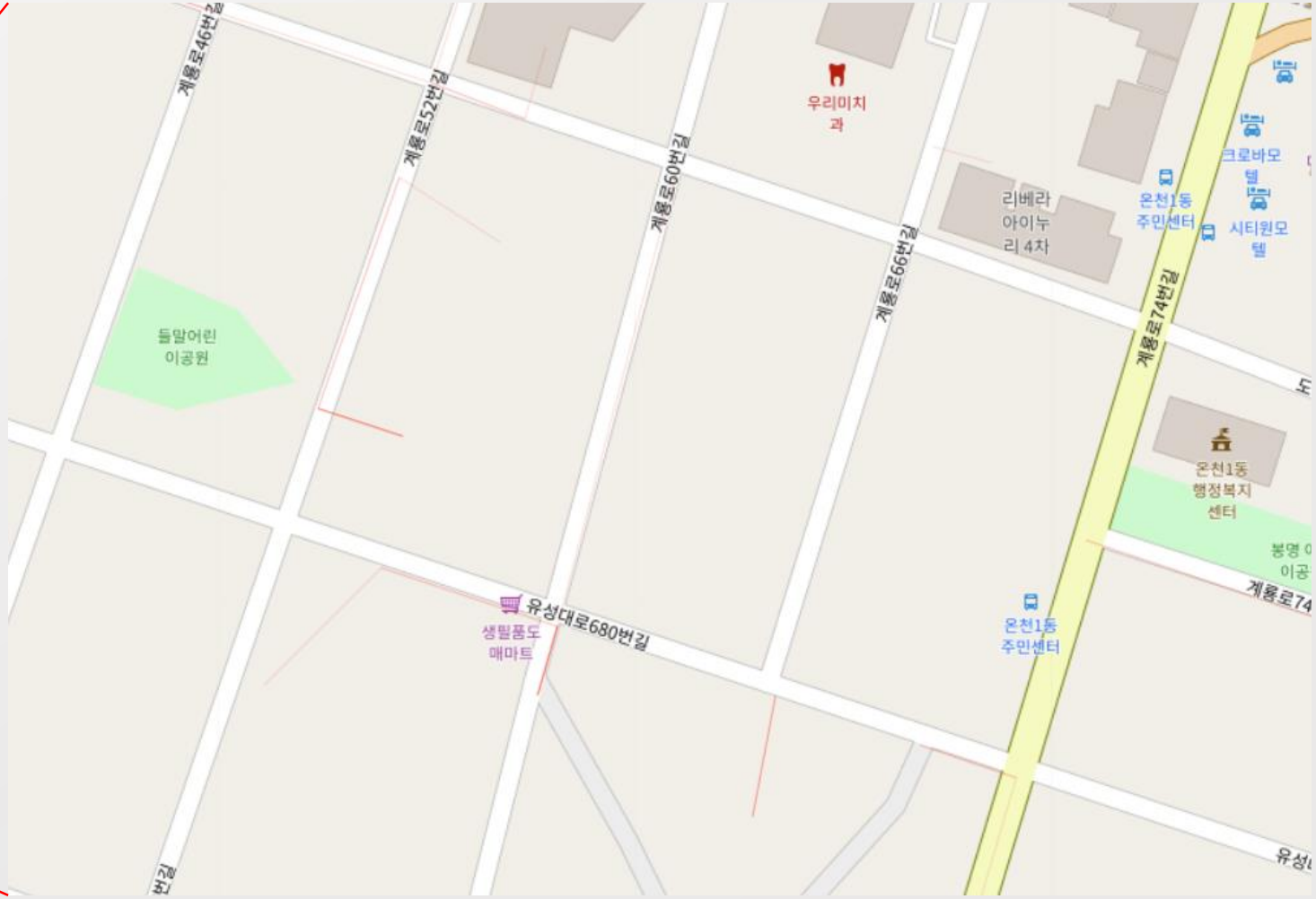
time, Simulation time



Gen_path

- 아직 잘 생성이 되지 않음
  - Sim_time은 torch.randint(0,24 ,shape)
    - 'KLEV': 3.2298613092226165
    - 'JSEV': 1.2653231065904282
    - 'nll_avg': 17.558022987127305
    - 'nll_min': 6.908400058746338
    - 'nll_max': 88.98231506347656

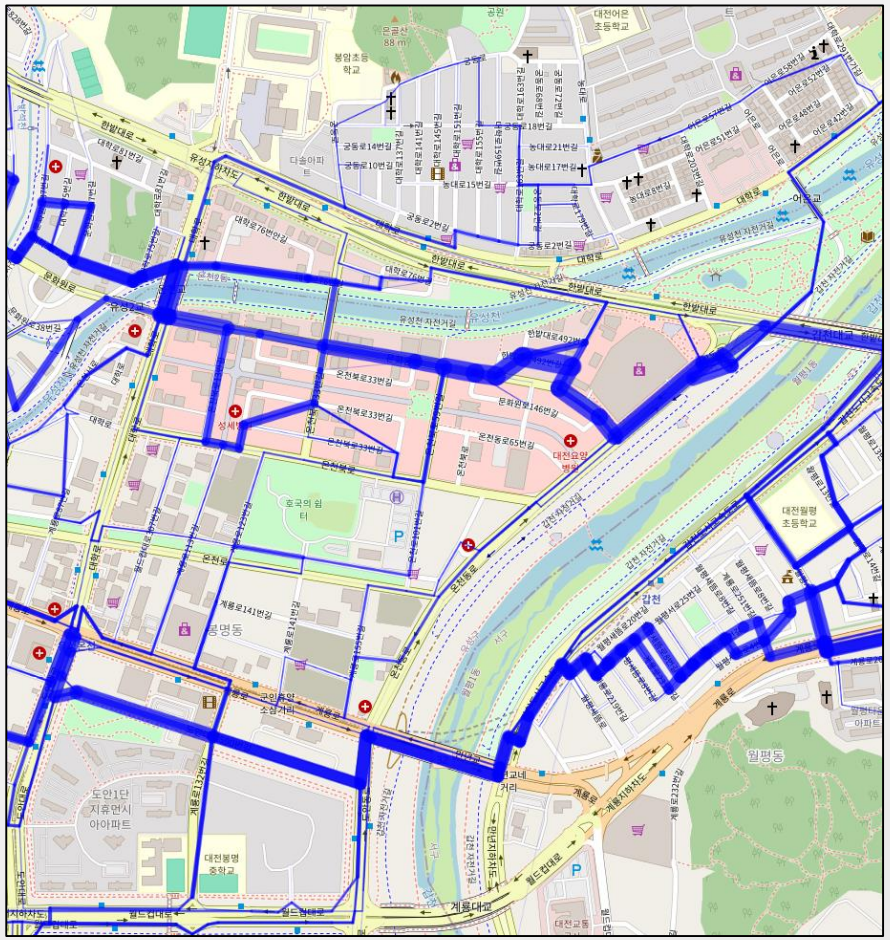




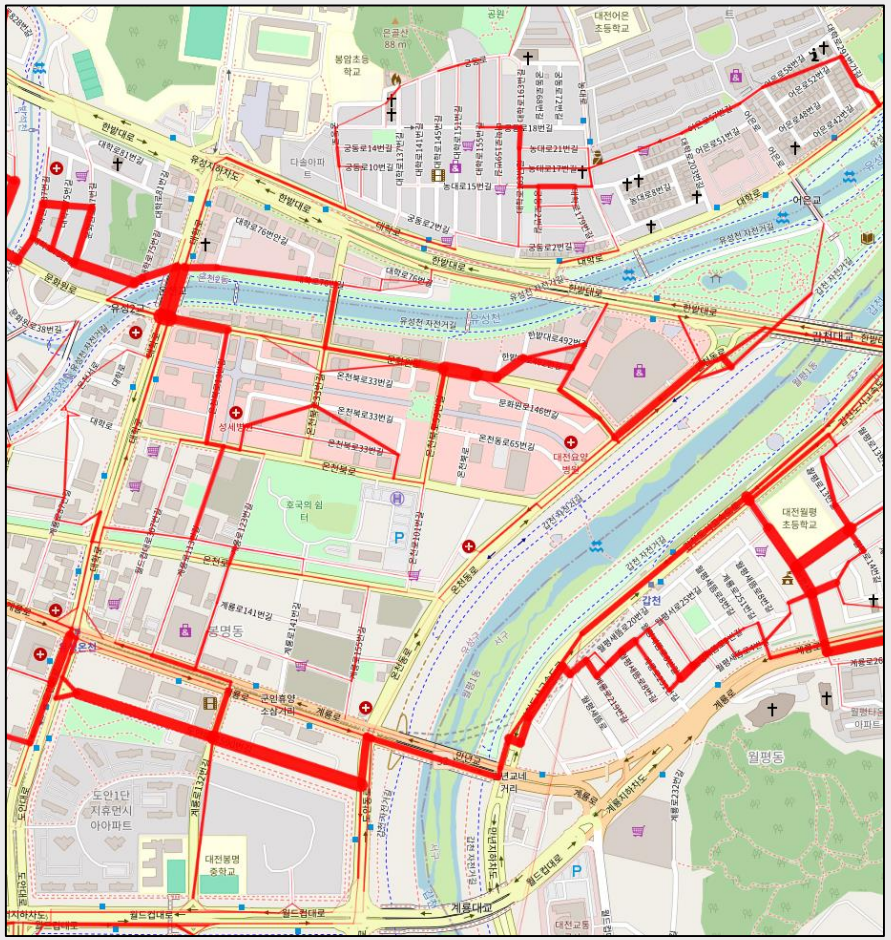
PREVIOUS SLIDES (25.02.05)

Road Heatmap by Generated Paths

- 학습 데이터 개수만큼 path generation
  ➔ 각 edge 를 지나는 path 의 수에 비례하여 굵기로 표현



Original Path



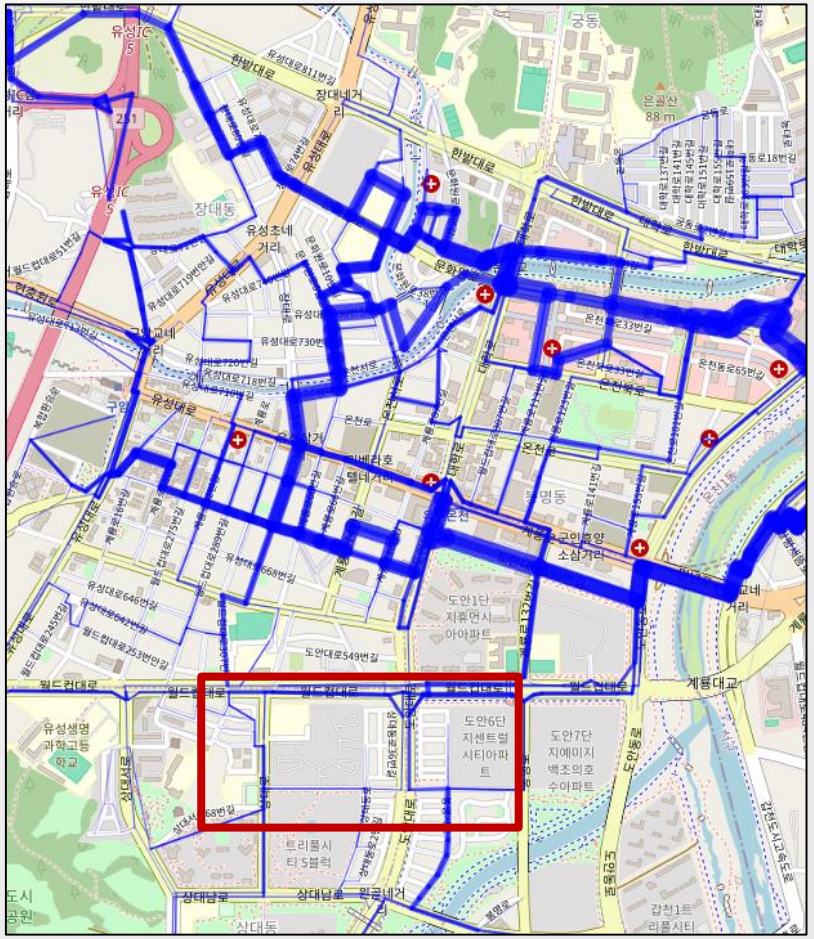
Generated Path

Application of Provided Road Network

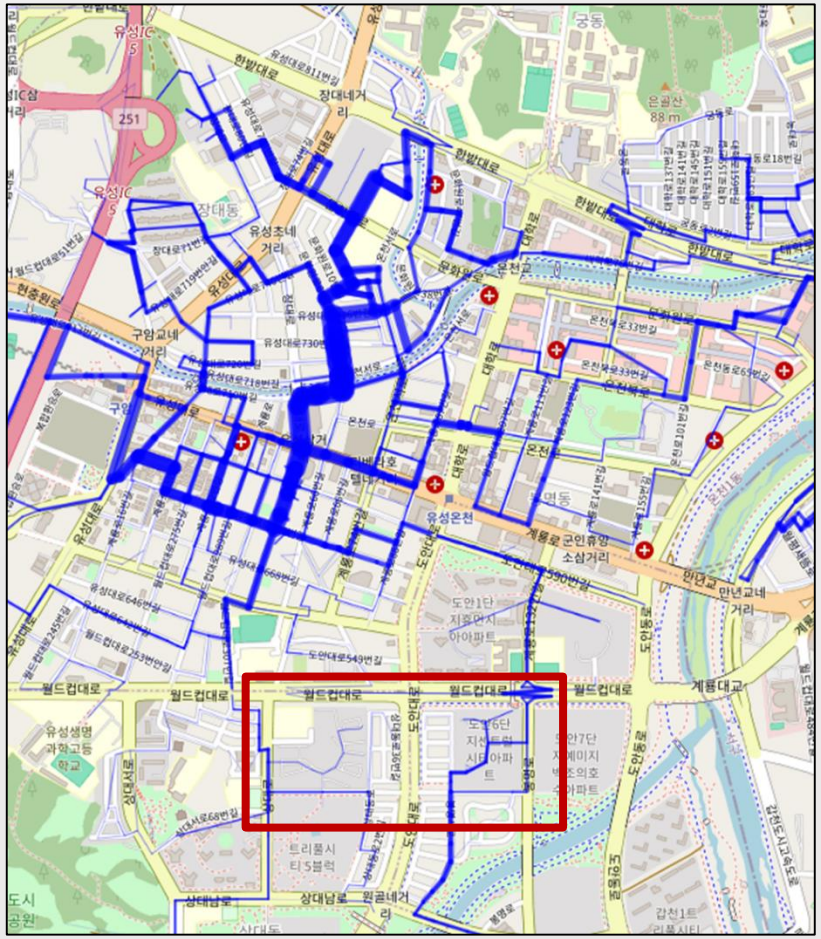
- 기존 그래프 구조는 기존 코드 구조대로 opensource data 인 OpenStreetMap 의 노드/엣지 정보를 활용

- ➔ ETRI에서 제공받은 road network 로 노드/엣지 정보가 반영되도록 코드 수정
  - 기존에 나타나지 않던 추가 노드 및 엣지를 확인할 수 있음 (빨간색 박스)

- Random 한 샘플이라 statistics가 다를 수 있음



Original path from OpenStreetMap



Original path from provided road network

Ongoing Tasks

- 시간에 따른 교통량 예측을 위하여 diffusion model에 timestamp condition 추가
  - 현재 diffusion model 은 unconditional generation 만 수행
  - 시간을 조건으로 하는 conditional generation 수행

- 전체 (1년치) 데이터를 사용할 수 있도록 데이터 전처리
  - 기존 결과들은 1일치, 7일치 수준에서 모델 학습 진행
  - 현재 받은 데이터 (총 1년치) 를 모두 활용할 수 있도록 데이터 전처리 진행

- 생성 데이터의 utility 검증 실험
  - 생성 데이터가 기존 데이터와 같은 분석 효과를 나타내는지 확인하는 정량적 지표를 내는 실험
  - 앞선 Heatmap 결과에 대한 정량적 지표 제시

PREVIOUS SLIDES (24.11.13)

Data-Driven Path Generation using Diffusion Models

Path data

Diffusion Model

Generated Path

Path #	Timestamp	Longitude	Latitude
1	1724900800	36.370724	127.366112
1	1724900810	36.368582	127.367849
1	1724900820	36.369701	127.370638
1	1724900830	36.371329	127.373306
2	1729872600	36.322684	127.423052
2	1729872610	36.323324	127.425090
…	…	…	…

각 Path에 대한 시간별 위도/경도 데이터

Reference Paper

- Shi et al., Graph-Constrained Diffusion for End-to-End Path Planning, ICLR 2024
  - Propose a diffusion-based model for end-to-end data-driven path planning
  - GDP: Graph-constrained Diffusion for Planning

Published as a conference paper at ICLR 2024

GRAPH-CONSTRAINED DIFFUSION FOR END-TO-END PATH PLANNING

Dingyuan Shi
Beihang University
chnsdy@buaa.edu.cn

Yongxin Tong
Beihang University
yxtong@buaa.edu.cn

Zimu Zhou
City University of Hong Kong
zimuzhou@cityu.edu.hk

Ke Xu
Beihang University
kexu@buaa.edu.cn

Zheng Wang
Independent Researcher
wangzheng04@gmail.com

Jieping Ye
University of Michigan
jieping@gmail.com
jpye@umich.edu

- Paper: https://openreview.net/forum?id=vuK8MhVtuu&noteId=T8PJSPZ75b
- Code: https://github.com/dingyuan-shi/Graph-Diffusion-Planning

Overview

- Solve the end-to-end path planning problem by converting it into a conditional sampling task.

- Determine the probability distribution of paths given an origin and destination.

$$\tilde{p}(x|ori, dst) = \underbrace{p_\theta(x)}_{\text{Unconditional path probability}} \underbrace{h(x|ori, dst)}_{\text{OD evidence probability}}$$

- So, they focus on two designs:
  - How to determine the **unconditional path probability**, i.e., $p_\theta(x)$?
    ➔ Graph-constrained diffusion models

  - How to incorporate the origin and destination data as **OD evidence probability**, i.e., $h(x|ori, dst)$?
    ➔ Autoregressive model under OD information

Categorical Diffusion Models

- $G =< V, E >$: a graph with its vertices & edges
- $P = (v_0, \dots, v_{|P|})$: a vertex sequence of path P
- v_t: categorical value at time t
- $q(\cdot)$: row vector of categorical probability
- Q_t: transition probability matrix
- $\boldsymbol{I}_{|V|}$: identity matrix with size $|V| \times |V|$
- $\mathbf{1}$: column vector with all elements set to 1

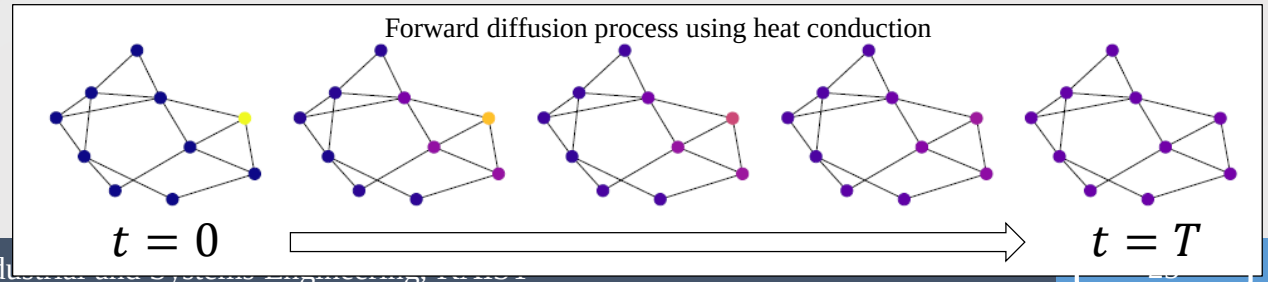
- Principles and requirements for diffusion process design
  - (Previous) Generic categorical diffusion models
    - $q(v_t|v_{t-1}) = \mathrm{Cat}(v_t; p = q(v_{t-1})Q_t)$
    - Prevalent choice for Q_t: $\alpha_t \boldsymbol{I}_{|V|} + (1-\alpha_t)\mathbf{1}\mathbf{1}^T/|V|$
    - Slightly perturbs the original vertex's probability and **uniformly redistributes** it across other vertices.
    - As t goes to infinity, the distribution of $q(v_t)$ will ultimately converge to a uniform distribution.
  - Graph-constrained situation
    - A generic categorical diffusion process **fails to capture the graph structure**.
    - We hope the forward diffusion process should exhibit **locality**.
    - Spread probabilities across **neighboring** vertices, converging to uniformity as t goes to infinity.
  - Requirements
    - Closed form for forward process $q(v_t|v_0)$
    - Computationally feasible posterior $q(v_{t-1}|v_t, v_0)$
    - Independence between prior distribution $q(v_T)$ and data distribution $q(v_0)$
    - Locality for smaller timestep t

Graph-constrained Diffusion Models

- Diffusion process for a single vertex
  - The heat conduction on Graph
    - $\frac{\partial h}{\partial t} = \Delta h = h(A - D)$ ➔ $h_T = h_0 C_t$ where $C_t = \exp\big((A - D)t\big)$
      - Considering each vertex as a point heat source.
      - h is a row vector of size $|V|$, indicating each vertex's heat.
      - Δ is the Laplacian operator, the sum of differences between it and its neighbors.
    - This equation suggests a vertex's heat change rate **depends on its heat difference with neighbors**.
    - We can express this using the **adjacency matrix $\boldsymbol{A}$ and degree matrix $\boldsymbol{D}$**.
  - Forward diffusion process
    - $q(v_t|v_{t-1}) = \mathrm{Cat}\big(v_t; p = q(v_{t-1})C_{\beta_t}\big)$ where $C_{\beta_t} = \exp\big((A - D)\beta_t\big)$
    - ➔ $q(v_t|v_0) = \mathrm{Cat}(v_t; p = \boldsymbol{v_0}\bar{C}_t)$ where $\bar{C}_t = C_{\beta_1} \dots C_{\beta_t} = C_{\sum \beta_i}$, $\boldsymbol{v_0}$: one-hot row vector indicating the vertex.
  - Reverse diffusion process
    - Posterior probability distribution is tractable: $q(v_{t-1}|v_t, v_0) = \mathrm{Cat}\big(v_{t-1}; p \propto \boldsymbol{v_t} C_{\beta_t} \odot \boldsymbol{v_0}\bar{C}_{t-1}\big)$
    - ➔ we train the network to estimate v_0.

<Properties of C_t>

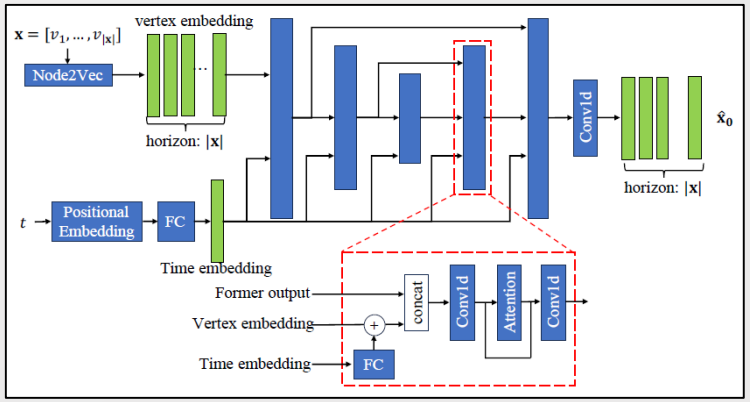
- $C_t = C_t^T$
- $C_{t_1+t_2} = C_{t_1}C_{t_2}$
- $C_{kt} = (C_t)^k$
- $C_t \to 11^T/|V|$ as $t \to \infty$ if graph is connected.
- $C_t \to I$ as $t \to 0$
- Summation of each row or column of C_t is 1.



Graph-constrained Diffusion Models

- Diffusion process for a path
  - Extend from the diffusion process of a single vertex to path.
    - Diffuse each vertex independently (rather than strictly maintain vertex connectivity)
  - Forward diffusion process
    - $q(x_t|x_0) = \prod_{i=1}^{|x|} q(x_t^i|x_0^i) = \otimes_{i=1}^{|x|} \mathrm{Cat}(x_t^i; p = \boldsymbol{x_0^i}\bar{C}_t)$
  - Reverse diffusion process
    - $q(x_{t-1}|x_t, x_0) = \prod_{i=1}^{|x|} q(x_{t-1}^i|x_t^i, x_0^i) = \otimes_{i=1}^{|x|} \mathrm{Cat}(x_{t-1}^i; p \propto \boldsymbol{x_t^i} C_{\beta_t} \odot \boldsymbol{x_0^i}\bar{C}_{t-1})$
    - ➔ We train the network nn_θ to estimate x_0.

- x: path, i.e., sequences of vertices
- x_t^i: i-th vertex in path x at diffusion timestep t
  - Subscript: diffusion timesteps
  - Superscript: vertex order in paths



Neural network structure for diffusion models

Algorithm 1: Training

Input: Dataset $\mathcal{P}$, model nn_θ
Output: model nn_θ

```
for training_steps ← 1, 2, ... do
    x_0 ~ p_data(x_0)
    t ~ U[1, T]
    x_t ~ q(x_t|x_0) by Eq.(8)
    p_θ(x̂_0|x_t) = nn_θ(x_t, t)
    calculate loss and update nn_θ
end
return nn_θ
```

Algorithm 2: Sampling

Input: nn_θ
Output: A generated path x.

```
Sample length l from Gaussian mixture model
x_T ~ ⊗_1^l U[1, |V|]
for t ← T, ..., 1 do
    x̂_0 ← nn_θ(x_t, t)
    sample x_{t-1} by Eq.(9)
end
x_0 ← apply beam search via x̂_0
return x_0
```

- Sample path sequence length
- Generate a random vertex sequence
- Perform the reverse process to denoise the random vertex
- Apply beam search to ensure connectivity

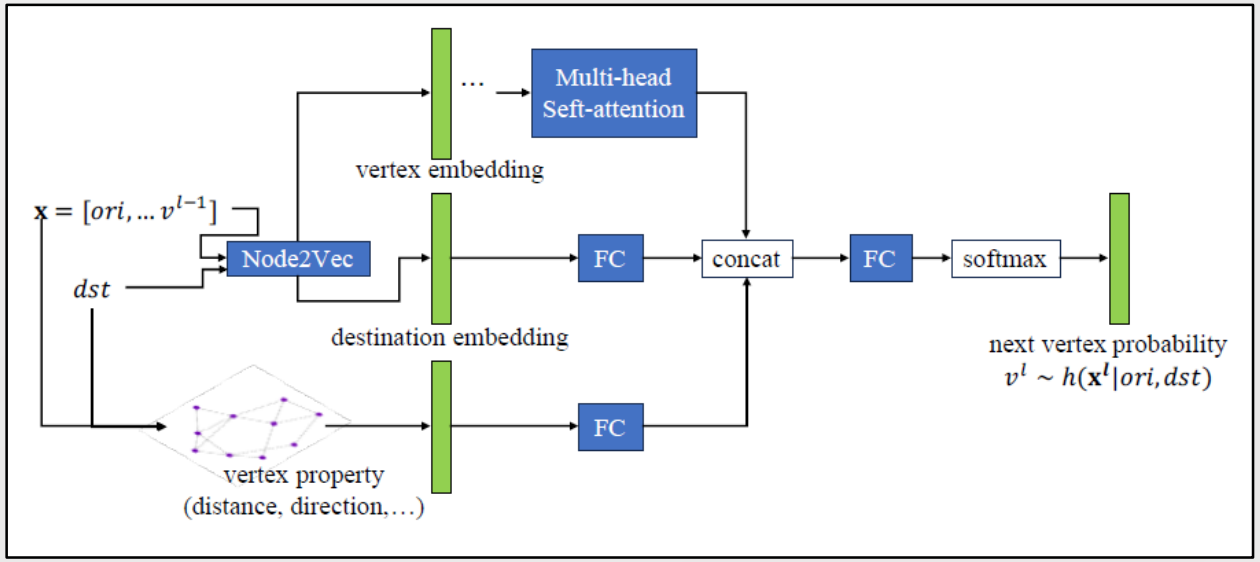
Path planning

- With the trained model, we can get an unconditional sample.
- How to get a conditional sample, with conditioned on ori, dst
  - $\tilde{p}(\mathbf{x}) = p_\theta(\mathbf{x})h(\mathbf{x}|ori, dst)$
- The simplest way is adding the ori, dst vertex information as a condition, but it hardly works.
  - Path planning has a strong spatial property.
- They propose a conditional generation process through two steps.
  1. Build spatial features for spatial property
  2. Integrate the OD evidence into unconditional sampling

Path planning

$$\tilde{p}(\mathbf{x}) = p_\theta(\mathbf{x})h(\mathbf{x}|ori, dst)$$

1. Build spatial features for spatial property
   - Input: [ori, conditional generated path (= $\mathbf{x}^l$)], dst
     - First input is [ori], dst
   1. Convert the [ori, conditional generated path], dst into
      - Embeddings via Node2Vec
        - Apply a multi-head self-attention to embeddings of [ori, conditional generated path]
          - To capture the sequence information.
      - Extract two spatial features
        - Distance feature : Distance from the current vertex to the dst based on coordinates
        - Direction feature : Cosine of angles between directions 1) towards dst from the current vertex and 2) from its adjacent vertices
   2. Concatenate together and get the probabilities for the next vertex
   - Output: $h(\mathbf{x}^{l+1}|ori, dst)$



Neural network structure for planning

Path planning

$$\tilde{p}(\mathbf{x}) = p_\theta(\mathbf{x})h(\mathbf{x}|ori, dst)$$

2. Integrate the OD evidence into unconditional sampling
   - In sampling process, we can get $h(\mathbf{x}|ori, dst)$
   - Calculate unconditional probabilities of nodes based on the previous conditional sample
     - Window size
     - Increase exponentially in sampling step
   - Get conditional probabilities
     - Product $h(\mathbf{x}|ori, dst)$ to unconditional probability
   - Sample node based on the probability
   - End when sampled node is dst

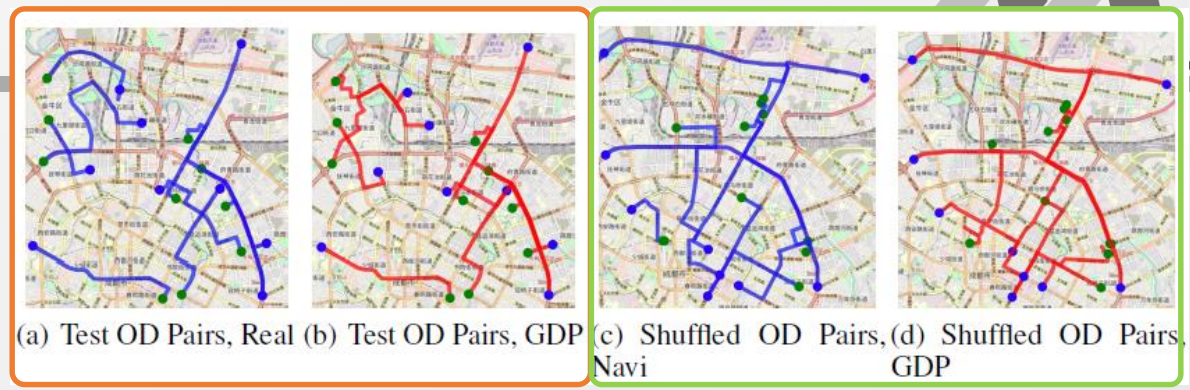
Algorithm 3: Planning

Input: nn_θ, seq_ϕ, ori, dst
Output: A generated planned path x.

```
x ← [ori]
window ← 1
idx ← window
while |x| < MAX_LEN do
    h_seq ← seq_φ(ori, dst, x)        // get evidence priori by sequence model
    if idx = window then
        x_T ~ q(x_T|x) by Eq.(8)                    // diffuse prefix sequence
        w ← ⊗_1^window U[1, |V|]          // random sequence with window size
        x'_T ← x_T||w          // concat prefix and random sampled sequence
        p_nn ← nn_θ(x'_T, T)[−window :]   // get probabilities for later use
        idx ← 0
        window ← window × 2       // increase window size exponentially
    end
    p̃_next ← h_seq ⊙ p_nn[idx]
    idx ← idx + 1
    x_next ~ p_next                                          // sample next vertex
    x ← x.append(x_next)
    if x_next = dst then break;
end
return x
```

Conditional planning algorithm

Experiments Basic setting



No shuffle → Ground Truth exists.

Shuffled → No Ground Truth.
Navigation data as Ground truth

- Dataset
  - Raw datasets contains GPS trajectories.
  - Shuffled OD pairs for validation
- Baselines
  - For path planning,
    - Dijkstra : Given OD path, search shortest path.
    - NMLR : distribution based on Markov property and search the path with high prob.
    - Key Segment(KS) : Given OD path, detects the relay vertex with the largest prob.
    - Navigation API : planning API
  - For path generation
    - N-gram : estimates the transition probability by counting the frequency.
    - HMM : Hidden Markov model
    - CSSRNN and MTNet : LSTM-based model for capturing long sequence pattern.
- Evaluation Metrics

For Planning	For Generation
Dynamic Time Wrapping (DTW) ↓ Longest Common Sequence (LCS) ↑	NLL (negative log likelihood) KL divergence w.r.t Edge visit distribution JS divergence w.r.t Edge visit distribution

Experiments Conditional Evaluation Result

- Evaluations for path planning

City	Metrics	Methods				
		DA	NMLR	KS	Navi	GDP (Ours)
A	DTW ↓	2.434	9.109	2.405	2.164	**1.683**
	LCS ↑	9.254	9.772	11.485	13.505	**16.730**
B	DTW	2.883	10.114	2.936	2.792	**2.426**
	LCS	9.233	10.066	12.207	14.390	**16.856**

City	Metrics	Methods				
		DA	NMLR	KS	Navi	GDP (Ours)
A	DTW	–	13.3	82.3	81.3	**86.6**
	LCS	–	70.3	83.8	79.0	**86.8**
B	DTW	–	11.1	76.9	78.7	**81.3**
	LCS	–	72.3	84.2	83.7	**87.3**

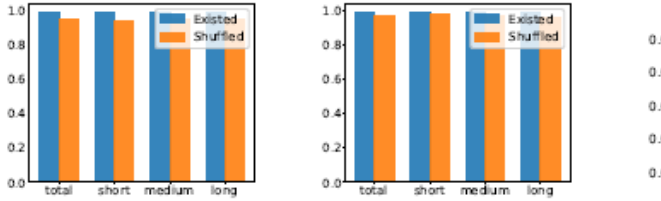
[Evaluations for path planning]

- DTW(km) : the difference between planned paths and ground truth path.
- LCS : the similarity between planned paths and ground truth path.

[Proportion of test cases better than Dijkstra]

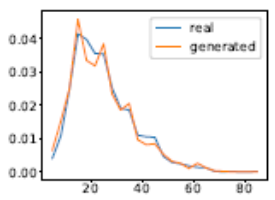
The result show robustness of performances.
It means that GDP can outperform Dijkstra's algorithm baseline.

- Illegal Cases detection

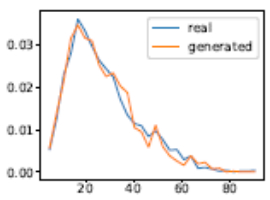


(a)

(b)



(a) City A



(b) City B

- Hit ratio over 94.2% ~ 99% in average.
- Histograms of path lengths of real and generated data are consistent.

Experiments Unconditional Generation Evaluation

- Comparison between real and generated path

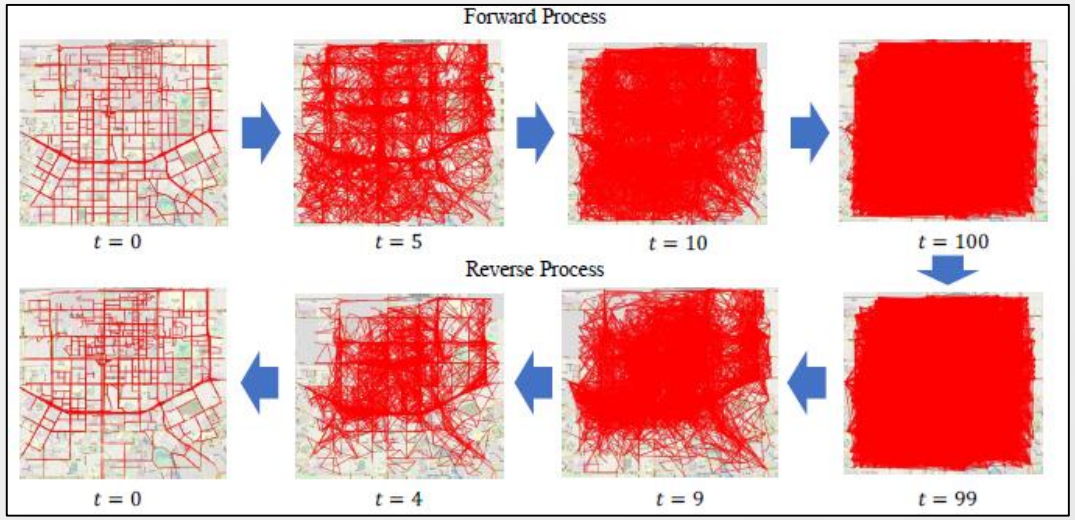
City	Metrics	Methods N-gram	HMM	CSSRNN	MTNet	GDP (Ours)	*GDP-
A	NLL ↓	300.144	278.238	27.742	69.859	**24.697**	43.582
	KLEV ↓	7.300	6.804	7.348	6.274	**5.724**	7.273
	JSEV ↓	3.309	3.061	3.3325	2.795	**2.518**	3.295
B	NLL	319.662	302.506	27.890	83.679	**23.740**	47.418
	KLEV	7.719	6.863	7.356	6.419	**6.128**	7.678
	JSEV	3.521	3.092	3.337	2.869	**2.724**	3.500

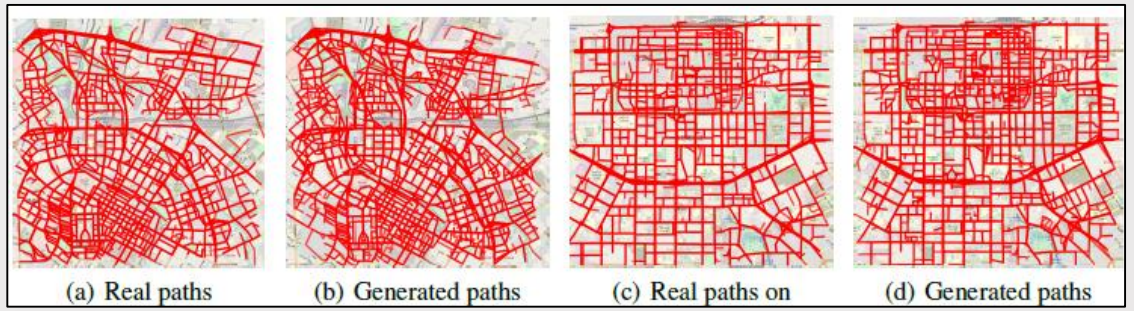
* : GDP without heat conduction

[Evaluations for unconditional generation]

- Count-based models(ex. N-gram, HMM) are based on Markov property, and it can hardly capture high order information. (NLL is high order evaluation metric)
- As sequence models do, GDP can capture the long sequence. information.(KLEV, JSEV is first order evaluation metric.)

- Visualization of how diffusion process works





Unconditional Generation using diffusion

Experiments on Daejeon Data Data Description

- 시나리오 생성을 위한 통계 기반 원본 데이터 (ETRI)
  1. GPS 데이터
     - 대전 택시/화물차/버스 등에 설치된 DTG 단말로부터 수집
     - 2019.07~2019.12, 2021.01~2021.06, 총 1년
       - 하루 단위로 분리
       - 각 날짜에 기록 시간, 차량 아이디, 위/경도, 속도 총 5개 항목 기록
     - 주요 통계량
       - 총 데이터 용량: 51.33 GB
       - 총 기록 횟수: 약 11억 4천만 번
       - 총 기록 차량 수: 2,859 대
       - 일당 평균 기록 횟수: 약 1,350만 번/일
       - 차량당 평균 기록 횟수: 약 23만 번
  2. 대전 지역 도로 교통망 데이터
     - 도로 교통망을 graph structure로 모델링 하기 위한 node와 edge 정보 등 기록
       - node id, 위/경도 정보
       - edge id, edge from/to 두 node id, edge shape 등의 형태
     - 주요 통계량
       - 총 node 개수: 40,887
       - 총 edge 개수: 99,741

timestamp	vehicleid	latitude	longitude	speed
20190701172918	1	36.35223	127.3704	25
20190701172919	1	36.35226	127.3703	21
20190701172920	1	36.3523	127.3702	20

GPS 데이터 예시

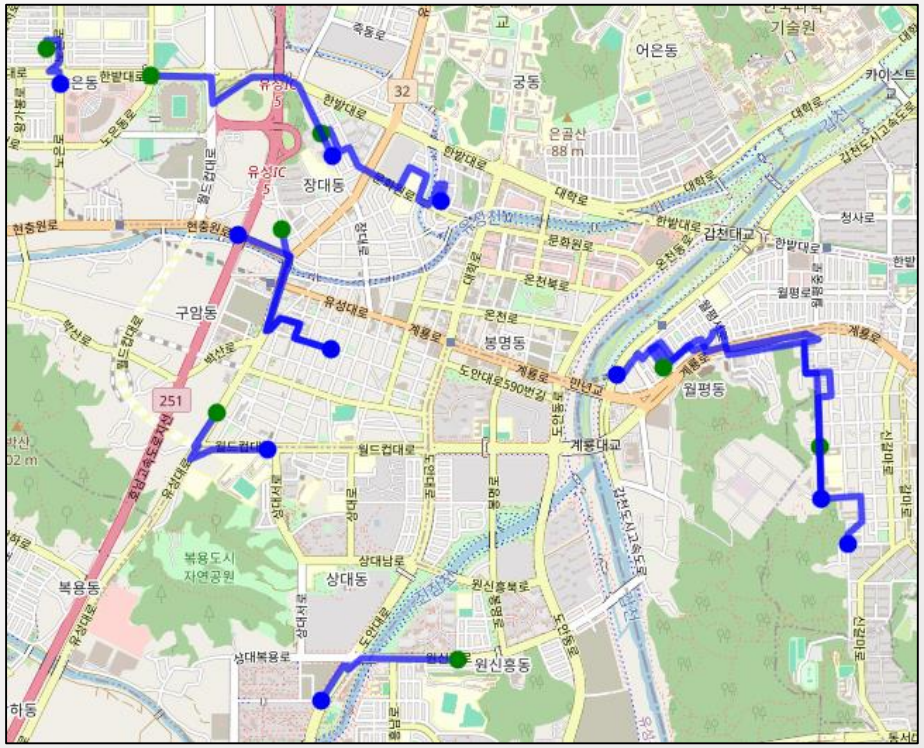
Edge id	From	To	numLanes	shape
-552800227	552800190	552800174	1	127.247893,36.290373 127.247845,36.290482 127.247791,36.290545
-553700011	553700123	553700025	1	127.295857,36.222533 127.2957,36.222534 127.295588,36.222536 127.295488,36.222548 127.29526,36.222595"
-553700012	553700124	553700025	1	127.294655,36.22295 127.294686,36.222933 127.294912,36.222756 127.294925,36.222737 127.29526,36.222595

도로 교통망 데이터 예시

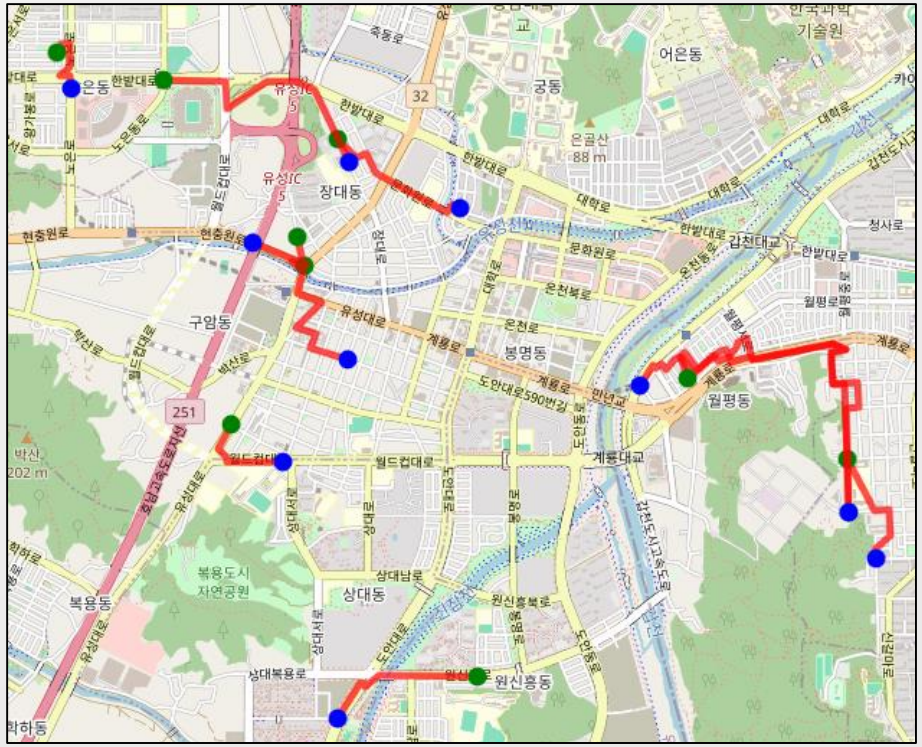
Experiments on Daejeon Data Experimental Result

- 2021.01.01~ 2021.01.07 데이터로 모델 학습하여 Path planning 진행
  - Path는 같은 차량 경로에서 임의로 150개씩 분리
  - Total Path: 230,016개

Original Path



Generated Path



Evaluation Results

Metric	Value
Mean DTW ↓	0.0035
Mean LCS ↑	8.577
Hit Ratio(median)	0.96

- DTW : the difference between planned paths and ground truth path.
- LCS : the similarity between planned paths and ground truth path.

Code Reproduce

- Github upload
  - https://github.com/AIxSIM/GDP_AAILAB