Journal of Data Science 19 (1), 126–141
January 2021

DOI: 10.6339/21-JDS1004
Data Science in Action

Large Scale GPS Trajectory Generation Using Map Based on Two Stage GAN

Xingrui Wang[1], Xinyu Liu[1], Ziteng Lu[1], and Hanfang Yang[1,*]

[1]*School of Statistics, Renmin University of China, Beijing, China*

Abstract

A large volume of trajectory data collected from human beings and vehicle mobility is highly sensitive due to privacy concerns. Therefore, generating synthetic and plausible trajectory data is pivotal in many location-based studies and applications. But existing LSTM-based methods are not suitable for modeling large-scale sequences due to gradient vanishing problem. Also, existing GAN-based methods are coarse-grained. Considering the trajectory's geographical and sequential features, we propose a map-based Two-Stage GAN method (TSG) to tackle the challenges above and 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 sequences. Discriminator conditioned on encoded map image restrains generated point sequences in case they deviate from corresponding road networks. Experiments on real-world data are conducted to prove the effectiveness of our model in preserving geographical features and hidden mobility patterns. Moreover, our generated trajectories not only indicate the distribution similarity but also show satisfying road network matching accuracy.

Keywords *generative adversarial network; GPS trajectory; spatial-temporal sequence*

1 Introduction

Various GPS sensors and mobile devices have been collecting massive location information, which has led considerable attention to spatial-temporal data like trajectory. Trajectory data has been studied for traffic jam analysis, trip pattern and behavior analysis, personalized route recommendation, etc. (Wang et al., 2013; Liu et al., 2012; Yue et al., 2009; Dai et al., 2015). Besides, in the field of self-driving, the synthetic trajectory is also widely used in trajectory planning, generating a set of likely trajectories for the future behavior of the obstacle and planning to produce safe action (Ferguson et al., 2008; Mohajerin and Rohani, 2019; Barth and Franke, 2009; Srikanth et al., 2019). However, large datasets remain difficult to obtain concerning privacy protection for companies and individuals through tremendous trajectory data produced in various ways. Hence, efforts have been made to create new trajectories with similar characteristics.

Most sequential trajectory synthesis models focused on predicting human or other object's mobility. In previous researches, sequential models like Markov Chain (Shokri et al., 2011; Baratchi et al., 2014) and Recurrent Neural Networks (Kulkarni and Garbinato, 2017; Gao et al.,

*Corresponding author. Email: hyang@ruc.edu.cn.

Received October 12, 2020; Accepted January 10, 2021

2017) were widely used because of their ability to capture hidden mobility patterns. However, without incorporating geographical information and relationships within neighboring trajectories, the performance of applying sequential models to a large targeting area is not satisfying. Generative models (Kulkarni et al., 2018) preserve features of the whole trajectory distribution better in terms of synthesizing large scale of trajectory data uniformly. Grid representation of trajectory point is used for input of GAN models (Ouyang et al., 2018), which limits the accuracy of generated trajectory since the precision level is determined by grid size. Besides, existing models preserve the geographical features of trajectories only by learning from given training data, which lacks specific local road network information.

In this paper, we propose a two-stage GAN method (Andreini et al., 2019; Duan, 2017) to generate vehicle trajectory that has similar distribution with the original dataset and also matches with road networks (as shown in Figure 1). First, we discretize trajectory data into grid representation and get the synthetic trajectory in grid form through a generative adversary network model. Then, to concretize the trajectory inside each grid, we append the map information in the form of map images as a condition in our second stage GAN. We adopt a fully convolutional network for encoding map images as an encoder and two parallel LSTM networks as the decoder or generator. The discriminator is also conditioned with the output of the encoder. The effectiveness of encoder in capturing road information in the image and the remarkable performance of decoder in generating trajectory sequence, along with an entry point and an exit point provided by the previous GAN model, makes it feasible to obtain the specific data points inside each grid.

In our model, we utilize the ability of generative models to preserve geographical features, relative features, and advantages of sequential generating model in learning from previous states conditioned on a given map image. The rest of the paper is organized as follows. Related work is presented in Section 2. In Section 3, we describe our proposed method. The experiment and evaluation are reported in Section 4. We conclude the paper by mentioning potential future work in Section 5.

2 Related Work

With the high demand for large trajectory datasets, various generative models have been designed to produce massive similar data while protecting privacy of the original dataset. Due to the special form of spatial-temporal data, Li et al. (2018) divide the targeting area into grids and transfer two-dimensional location data into a one-dimensional grid number series. In this way, trajectory points inside the same grid are treated as the same. The synthetic trajectories are concatenated from the decomposed original dataset. It preserved statistical features of trajectory length and speed but failed to capture geographical features in traces. Sequential models like Markov Chain were also widely used, and they model each location as a state (Baratchi et al., 2014; Shokri et al., 2011). Ouyang et al. (2018) also transformed a trajectory into a sequence of stays, where the time and duration of each stay in each grid were recorded in the corresponding matrix. Grid representation improved efficiency in processing trajectory data but lost precision when increasing grid size.

In the case of complex generative models, deep learning models and GANs are highly effective learning methods that have been widely used nowadays. Apart from image data, time-series data like mobility data can also be generated by GANs. Kulkarni et al. (2018) introduced four RNN and GANs models as benchmarks in mobility data generation and compared their

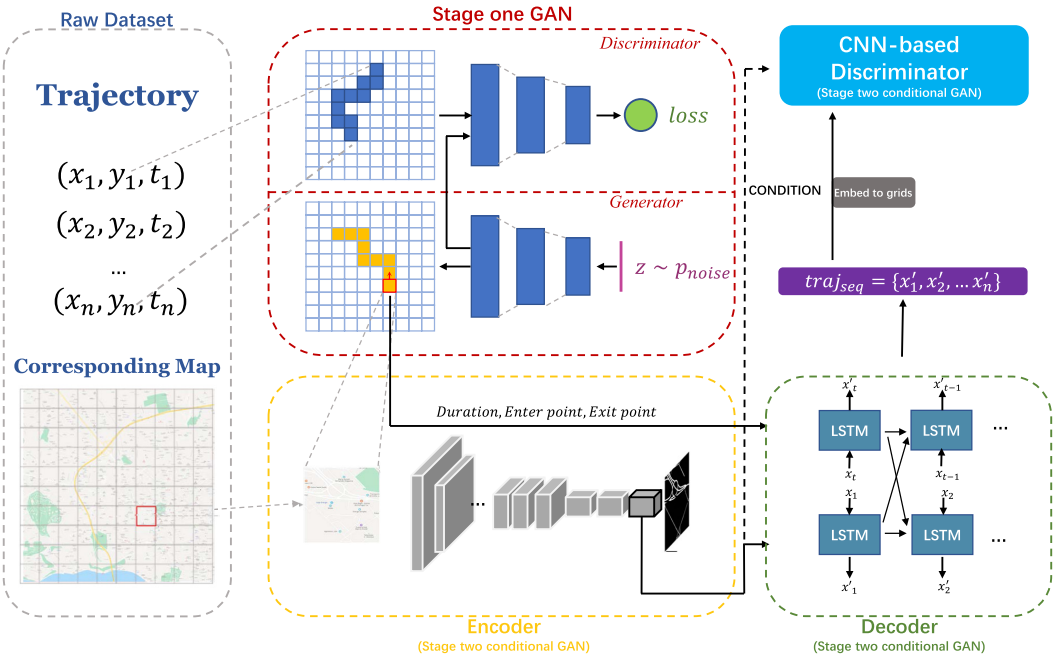


Figure 1: The framework of our step-by-step model. In the training phase for the first stage, trajectory data are transformed into grid representation and then fed into CNN layers. In the second stage, the encoder, decoder, and discriminator are trained together, where the input are map image, corresponding entry point and exit point gained from the first stage. The algorithm to connect the two steps is illustrated in Section 3.3.

differences in terms of geographic and semantic similarity, statistical similarity, long-range dependencies, and privacy test, and they achieved slightly different results in evaluation for their structural distinctions. Baratchi et al. (2014) designed a sequential data generating model based on Long-Short-Term Memory (LSTM) architecture. The model infers the whole sequence by capturing the previous location states information and transition features. More recently, Ouyang et al. (2018) pointed out the problem of exposure bias when generating samples by self-unrolling in the previous model, and then proposed a non-parametric generative model which is trained with the loss function from WGAN-GP (Gulrajani et al., 2017). It outperformed sequential models through combining geographical and semantic features, but it suffers from precision loss due to the limitation of grid representation. Still, the generative adversary network shows an impressive performance in detecting local patterns and learning the joint distribution of original trajectories.

The encoder-decoder framework is widely used in deep learning tasks (Vinyals et al., 2015; Xu et al., 2015; Cho et al., 2014; Garg et al., 2019; Gehring et al., 2017; Lu et al., 2017; Wu et al., 2016a,b). In some sequence-to-sequence tasks, the encoder-decoder is exploited to extract the prudential feature vector from the input sequence and decode it to the output sequence, such as in machine translation (Cho et al., 2014; Garg et al., 2019; Wu et al., 2016b). In imaging caption (Vinyals et al., 2015; Xu et al., 2015; Lu et al., 2017; Wu et al., 2016a), the encoder transfers images to contextual information and allows the decoder to generate descriptions about images. In this work, we combine convolutional nets for semantic segmentation with RNN for sequence modeling. The deep convolutional network is designed to encode map images to contextual

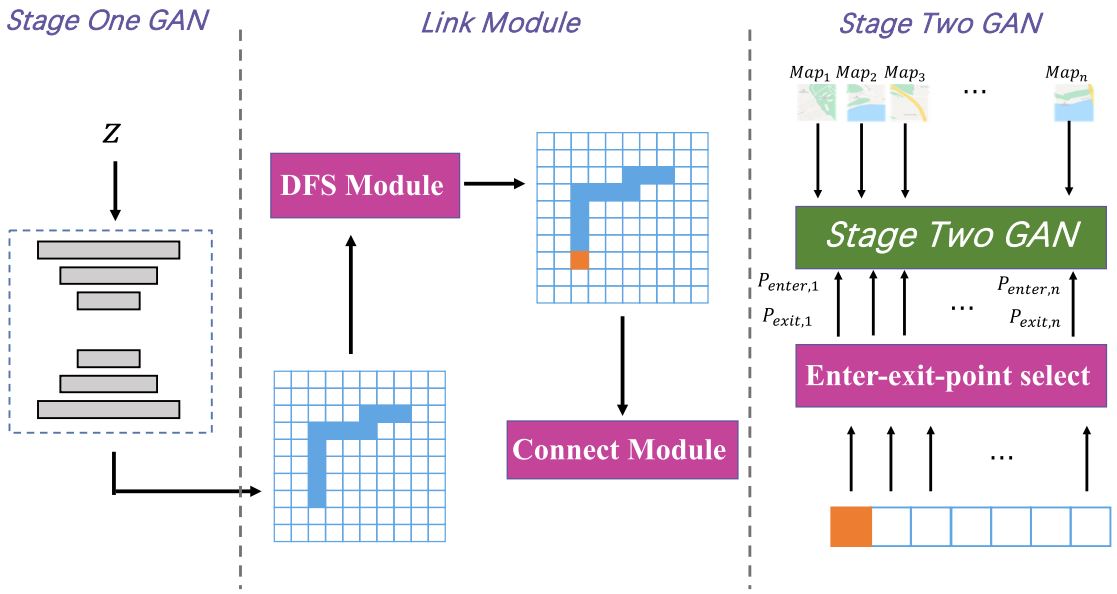


Figure 2: The pipeline of two stage GAN (TSG). In connection part, DFS (depth-first search) is applied to find a starting grid and connect module is to serialize the matrix-form trajectory data to a sequence-form one. Enter-exit point selection module is to obtain the prior information, enter point and exit point for next step LSTM based on map image.

features, and the RNN is trained to combine map features and trajectory information to infer the following trace points.

3 Approaches

The whole architecture follows a coarse-to-fine pattern and is divided into three parts. A normal GAN to generate plausible grids representation trajectory matrix on a coarse-grained level, and a novel map-conditioned GAN are designed to generate fine-grained coordinate of trajectory point inside grids. A connection module to link the corresponding result of the two stages (see Figure 2).

Notation Let $\text{Traj}_{\text{seq}} = \{p_{t_1}, p_{t_2}, \ldots, p_{t_n}\}$ be origin trajectory data, and $p_{t_i} = (x_i, y_i, t_i)$, where $i = 1, 2, \ldots, n$ is the GPS sample point with coordinates (x_i, y_i) in time step t_i. In our first step GAN, we discretize the region into a $N_1 \times N_2$ grid and each trace points falls into a grid based on its location. To record the trajectory information in a grid matrix $\text{Traj}_{\text{mat}} \in \mathbb{R}^{N_1 \times N_2}$, each element in the matrix $d_{i,j}$, where $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, N_2$ denote the duration time of trajectory in the specific grid. Given synthetic trajectory data in grid representation, the second stage GAN is to transfer data in matrix $\widehat{\text{Traj}}_{\text{mat}} \in \mathbb{R}^{N_1 \times N_2}$ into original GPS points sequence in length n: $\widehat{\text{Traj}}_{\text{seq}} = \{\widehat{p}_{t_1}, \widehat{p}_{t_2}, \ldots, \widehat{p}_{t_n}\}$ by a fine prediction.

3.1 Stage One GAN: With Grid Representation

Generative Adversarial Network is a framework to train generative models via an adversarial process. We simultaneously train two models: a generative model G that captures the data

distribution, and a discriminative model D that estimates the probability that the sample came from the training data rather than G (Goodfellow et al., 2014).

In stage one, trajectory in grid representation $\text{Traj}_{\text{mat}} \in \mathbb{R}^{N_1 \times N_2}$ is the input. The goal in the training procedure is to minimize the distance between the joint distribution of training trajectory data P_r and synthetic data distribution P_g. The loss function in WGAN-GP (Gulrajani et al., 2017) adopts Wasserstein distance to specify the correct optimization goal and gradient penalty to avoid gradient vanishing and gradient exploding. The loss function is

$$L = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}\big[D(\tilde{x})\big] - \mathbb{E}_{x\sim\mathbb{P}_r}\big[D(x)\big] + \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\big[\big(\|\nabla_{\hat{x}}D(\hat{x})\| - 1\big)^2\big], \tag{1}$$

where $\mathbb{P}_{\hat{x}}$ is sampling uniformly along straight lines between pairs of points sampled from the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$. D denotes the discriminator.

3.2 Stage Two Map-Conditioned GAN: Inside Grid

This part is designed to generate the exact coordinates of the trajectory points inside grids. Given a certain pair of enter point (x_1, y_1) and exit point (x_t, y_t), we need to generate a trajectory as a sequence of points $\{(x_1, y_1), \ldots, (x_t, y_t)\}$ that looks realistic when placed in a corresponding map patch. The corresponding map needs to be encoded into a latent vector, that can be used as a condition in our second stage GAN. The generator decodes the information step by step with the help of a recurrent neural network (Hochreiter and Schmidhuber, 1997). We noticed that the task of generator shared some common grounds with neural image caption (Vinyals et al., 2015), but there are three main differences. First, this problem needs us to make predictions in a continuous space which is a regression problem. Second, instead of starting from a *<start>* token and generating words step by step until a *<end>* token shows up, the trajectory synthesis problem needs us to consider both the entry and exit points from the very beginning and ensure the first several points not to take a wrong direction. Third, trajectory synthesis and image caption both employ information from images in different ways. But the latter one uses more semantic information because we expect the network to understand WHAT things in an image mean and we hope to get spatial information from images in order to figure out WHERE points in an image are. The role of discriminator here is to restrain the generated trajectory inside road networks by judging if it is plausible. We will describe our conditional GAN framework separately in this part.

3.2.1 Encoder

Our encoder is designed to extract the potential feature in the raw map images as the condition for the generator. Different from imaging caption task (Vinyals et al., 2015; Xu et al., 2015), we apply a simple fully connection architecture (Long et al., 2015) as the encoder, since the road network, the key feature of map images is similar to a simple segmentation mask rather than semantic information (Garg et al., 2019). Encoder takes the raw map image $I \in \mathbb{R}^{H\times W\times C}$ as input and obtains the road feature $F \in \mathbb{R}^{H'\times W'\times C'}$ (see Figure 1).

3.2.2 Generator/Decoder: Symmetric LSTM

Image as Condition All of the coordinates of point (x, y) have been normalized into $(-1, 1)$ before being used. We employ the same way as the neural image caption to make use of image information, i.e. transforming the encoded feature map from FCN into the initial hidden state

(a) (b) (c)

Figure 3: Road network information extracted from map images. (a) is the original map image. (b) is the map of every road in this area while (c) is the target feature we expect to extract from (a).

h_0, c_0 of LSTM cells through two separate fully connected layers. Moreover, we attach the entry and exit point embedding vector with encoded images latent vector at the same time. Therefore, the initial hidden state is calculated as:

$$h_0 = f_{init,h}\big(F \oplus E(p_{enter} \oplus p_{exit})\big), \tag{2}$$

$$c_0 = f_{init,c}\big(F \oplus E(p_{enter} \oplus p_{exit})\big), \tag{3}$$

where $F \in \mathbb{R}^{h \times w \times c}$ denotes the feature map extracted by encoder module and E is the position embedding layer (as shown in Figure 4). $f_{init,h}$ and $f_{inti,c}$ mean fully connected layers as initial layer for h, c respectively, $p_{enter} = (x_0, y_0)$ and $p_{exit} = (x_t, y_t)$ mean entry and exit point as we defined above. The $\oplus$ denotes the concatenation operation of tensor.

Symmetric LSTM We have tried vanilla LSTM, but we found exposure bias (Bengio et al., 2015) problem when making predictions from a single start point recurrently which results in points at the tail of the generated sequence deviating from the ground truth exit point. Ways to address this problem such as Scheduled Sampling have been proposed (Bengio et al., 2015). However, we noticed that our problem possesses a good property of symmetry, i.e. generating sequence from head to tail is equal to that in reverse order. Therefore, inspired by the structure of Social-LSTM (Alahi et al., 2016) and Bi-directional LSTM (Huang et al., 2015) we propose our decoder which consists of two LSTMs as shown in Figure 4.

In the training phase, these two LSTM process a sequence in a reverse order respectively using the so-called Teacher Forcing strategy. Meanwhile, hidden states are exchanged between two LSTM, which helps the LSTM to choose the right direction. Specifically, at time step t $(t = 1, \ldots, T)$,

$$\begin{pmatrix} \hat{x}_{t+1} \\ \hat{y}_{t+1} \\ h_t \\ c_t \end{pmatrix} = \text{LSTM} \begin{pmatrix} \hat{x}_t \\ \hat{y}_t \\ h_{t-1} \oplus h\_rev_{t-1} \\ c_{t-1} \oplus c\_rev_{t-1} \end{pmatrix}, \tag{4}$$

$$\begin{pmatrix} \hat{x}_{T-t} \\ \hat{y}_{T-t} \\ h_t \\ c_t \end{pmatrix} = \text{LSTM}_{\text{rev}} \begin{pmatrix} \hat{x}_{T-t+1} \\ \hat{y}_{T-t+1} \\ h\_rev_{t-1} \oplus h_{t-1} \\ c\_rev_{t-1} \oplus c_{t-1} \end{pmatrix}, \tag{5}$$

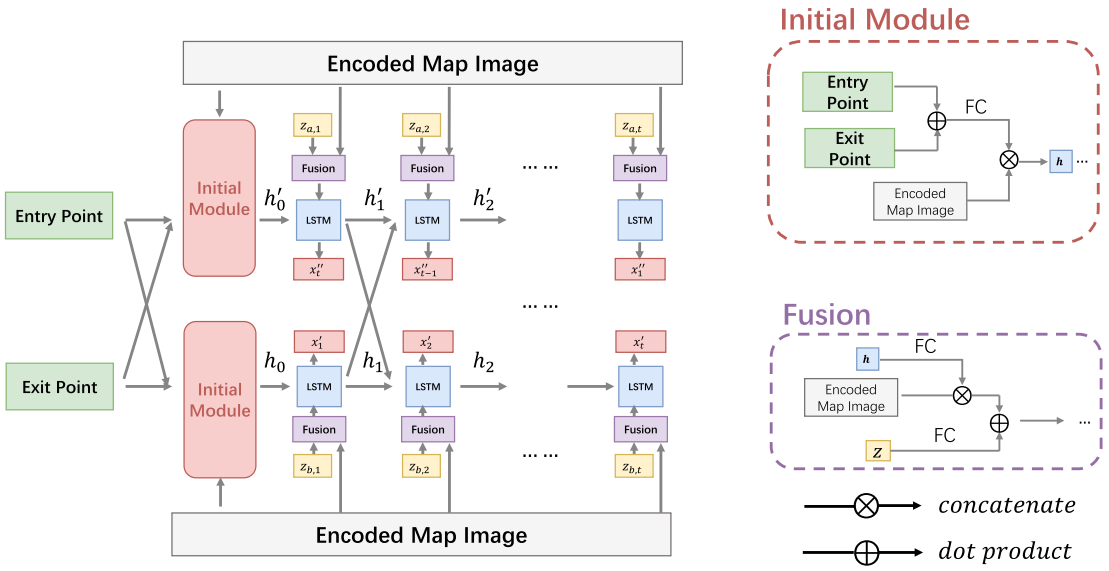


Figure 4: Generator/Decoder: Symmetric LSTM. Given an entry point, an exit point and an encoded image, we design two parallel LSTMs that respectively generate trajectory points sequence from entry to exit and from exit to entry. In the process, two parallel LSTMs share hidden states h_{ai}, h_{bi}. The final output is a weighted combination of two output sequences: $(x'_1, x'_2, \ldots, x'_n)$, $(x''_1, x''_2, \ldots, x''_n)$.

where LSTM and LSTM_{rev} represent for a single step of well-known Long Short-Term Memory framework (Hochreiter and Schmidhuber, 1997) and $\oplus$ denotes concatenation of vectors.

We combine ℓ_2 loss with the traditional objective function since the generator aims not only to fool the discriminator but also to predict tracepoints near the ground truth.

$$L_G = \mathbb{E}_{z,C}\big[\log\big(1 - D\big(C, G(z, C)\big)\big)\big] + \lambda \mathbb{E}_{x,z,C}\big[\|x - G(z, C)\|_2\big], \tag{6}$$

where x denotes ground truth trajectory, and C denotes condition which is the map feature in our case. w_D denotes the weight of loss from Discriminator, while λ denotes the weight of ℓ_2 loss. The generator takes latent variable z as input, and output synthetic trajectory $G(z, C)$.

3.2.3 CNN-based Discriminator

Inspired by Chen et al. (2019), we use a CNN-based discriminator in this sequence generation sub-task. Given a map image and a corresponding points sequence, the discriminator is trained to tell if the point sequence is plausible, which means it should match with road networks and also follow the pattern that real trajectories possess. Map image was previously encoded as $F \in \mathbb{R}^{H' \times W' \times C'}$. The target grid is divided into $H' \times W' \times 1$ sub-grids, and then the sequence of the points is transformed into a $H' \times W'$ trajectory matrix in the same way we did previously. Concatenate the encoded image and the trajectory matrix together as a $C' + 1$ channel input for convolutional layers.

The objective function of the discriminator is

$$L_D = \mathbb{E}_{z,C}\big[\log\big(1 - D\big(C, G(z, C)\big)\big)\big] + \mathbb{E}_{x,C}\big[\log D(x, C)\big]. \tag{7}$$

Algorithm 1 Connection algorithm.

Input: $\text{Traj}_{\text{mat}} \in \mathbb{R}^{N_1 \times N_2}$.
Parameter: N_1, N_2.
Output: $\text{Grid}_{\text{seq}} = (g_1, g_2, \ldots, g_m)$, $g_i = (x_i, y_i, t_{x_i,y_i})$.

1: Let Adjacency matrix $A = \text{Traj}[1]$.
2: Randomly pick $t \in A$ and $t \neq 0$ and apply Depth First Search (DFS).
3: Find $A[x_1][y_1] = t_{x_1,y_1}$ with DFS and set $g_1 = (x_1, y_1, t_{x_1,y_1})$.
4: Let $i = 1$.
5: **while** Neighbour(g_i) is not None **do**
6: Randomly pick $g_{i+1} \in \text{Neighbour}(g_i)$.
7: $g_{i+1} = (x_{i+1}, y_{i+1}, t_{x_{i+1},y_{i+1}})$.
8: **end while**
9: **return** Grid_{seq}

3.3 Link Module

In the first step, GAN generates multiple matrices that each of which represents a synthetic trajectory. Combining all these synthetic matrices can fully show the distribution feature of trajectories. For each generated matrix $\widehat{\text{Traj}}_{\text{seq}} = (t_{ij})$, where t_{ij} represents the duration time, we apply our connection algorithm 1 to transfer $\widehat{\text{Traj}}_{\text{seq}}$ to $\{\widehat{p}_{t_1}, \widehat{p}_{t_2}, \ldots, \widehat{p}_{t_n}\}$. According to the grid sequence, we can obtain an entry direction and an exit direction for each grid. Based on road networks, the exit point can be selected randomly from the boundary in the exit direction of the road network, and the entry point can be inferred from the previous exit point. As for the initial grid, we randomly select an entry point from the boundary of road networks. With an entry point, an exit point, and the duration time, the second part of our model can carry on the job to generate GPS points sequences inside specific grids.

4 Experiment

In this section, we present the experiment to evaluate the performance of our proposed model in generating trajectories with a given dataset. We evaluate the synthetic trajectories in terms of their similarity to real data. Besides, we also conducted ablation studies to test the effectiveness of certain parts of our model respectively.

4.1 Dataset

We use an accurate taxi trajectory dataset, in the city of Porto, in Portugal. The dataset is generated by 442 taxis, using mobile data terminals installed in the vehicles in a complete year (from 01/07/2013 to 30/06/2014), which has approximately 1.57 million trajectories. The dataset contains GPS point sequences with 15 seconds between every two points. Considering 15 seconds for a driving taxi can be quite a long distance, in order to get more detailed road information and shorten time interval, we adopt cubic interpolation to pre-process to get more dense data, with 3 seconds between every two points.

As for map images we used for the encoder, we take screenshots with Map API. We cut and adjust all the images to match the grid areas we chose in the targeting region.

4.2 Benchmark

- FTS-IP (Li et al., 2018): This model proposed a feature-preserving method to decompose and generate a new trajectory. By decomposing and concatenating given real trajectories, statistical features of length, speed and density are well-preserved.
- LSTM (Kulkarni and Garbinato, 2017): This model adopted the ability of RNN to learn and model problems over sequential data having long-term temporal dependencies. It completed a new trajectory by iteratively feeding the current output trajectory sequence as input to the next step to the trained model, starting at some arbitrary location.

4.3 Generated Trajectory Distribution

4.3.1 Overall Distribution

In our experiment, we generate 8500 trajectories with our model for evaluation. To evaluate the statistical similarity between real trajectory distribution and synthetic distribution, we choose Jensen Shannon Divergence (JSD) as the metric. JSD is also known as information radius (IRad) (Weikum, 2002), which is based on the Kullback–Leibler divergence. Because JSD is symmetric and finite, it is widely used in analyzing the performance of GAN in generating similar distribution. The JSD of two probability distributions P and Q is computed as:

$$\mathrm{JSD}(P||Q) = \frac{1}{2}\mathbb{E}_{\mathbb{P}}\left[\log\frac{P}{X}\right] + \frac{1}{2}\mathbb{E}_{\mathbb{Q}}\left[\log\frac{Q}{X}\right], \tag{8}$$

where $X = \frac{1}{2}(P + Q)$.

To evaluate the overall distribution of the synthetic data, we denote $p_o(r)$ is the probability that location r is visited.

4.3.2 Distribution of Length

We evaluate the distribution of 'length' in two different perspectives: sequence length and distance length (see Figure 6).

- $p_s(l)$ denotes the probability that the length of a trajectory sequence equals to l. Since the time interval between two points is fixed, $p_s(l)$ actually reflects the distribution of driving time of a trip.
- $p_d(l)$ denotes the probability that the actual distance of a trajectory equals to l. Distance length is calculated as $D = \sum_{i=1}^{n-1} \mathrm{dis}(p_i, p_{i+1})$ where $\mathrm{dis}(p_i, p_{i+1})$ is the geographical distance between p_i and p_{i+1}.

We calculate the distance length of each trajectory from both real data and synthesis data and plot the histograms (see Figure 6). In Table 1, we calculate the aggregate statistics, namely $p_o(r)$, $p_s(l)$, $p_d(l)$ from the synthesis set to the real set. TSG receives the smallest JSD score of overall distribution $p_o(r)$ and distance length distribution $p_d(l)$. Although LSTM obtains the smallest JSD score of $p_s(l)$, it cannot preserve enough geographic information, as the JSD of $p_o(r)$ is large.

The above results prove the capability of our generative model in preserving various useful properties of real data. But given the various driven patterns in real-world cases, there are

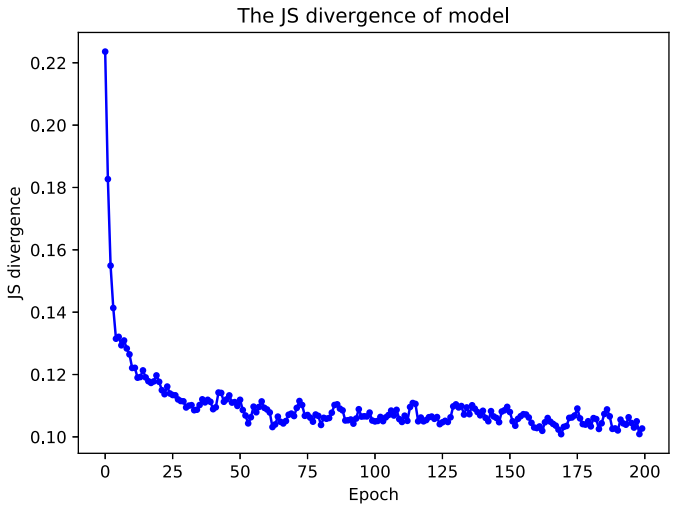
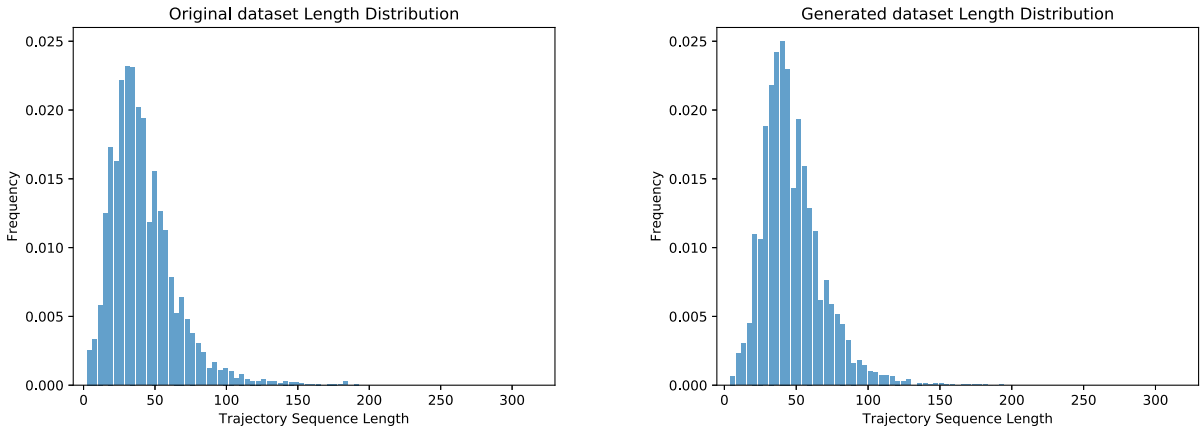


Figure 5: JSD convergence with respect to training epochs. The figure shows that the JSD converges to a relative small value after around 150 epochs.



(a) (b)

Figure 6: (a) and (b) show similar distribution of sequence length between original trajectories and generated trajectories.

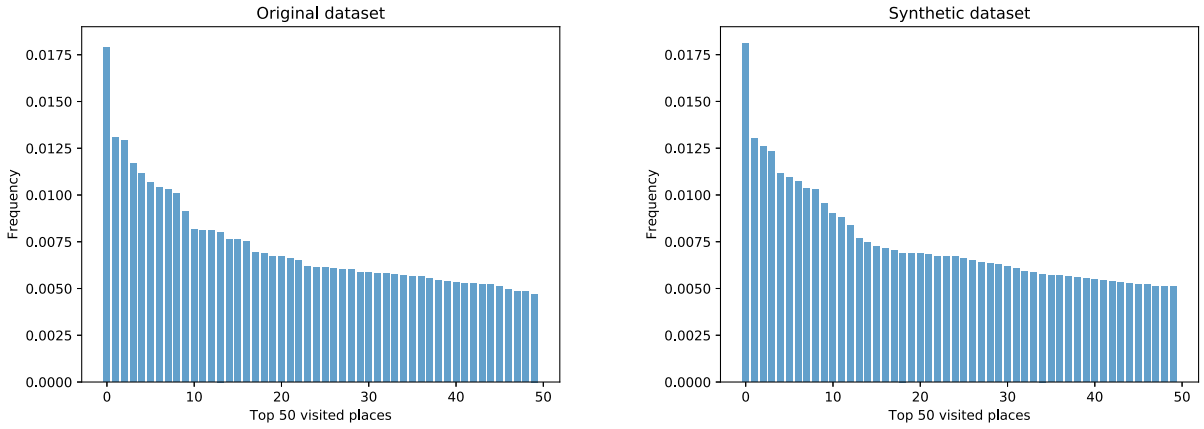
multiple types of trajectories. For instance, the trajectories of taxi and the ones of private car might have divergence in distribution of traces distance. It can be a possible cause for the right tail data in origin dataset which model does not cover.

4.3.3 Top *N* Visited Places

The overall distribution of vehicle trajectories in a specific area reflects the popularity of different locations, which is valuable in various data mining and machine learning tasks like location recommendation. We are now comparing the proportion of visits of the top 50 visited places among the original dataset and synthetic dataset (see Figure 7). Here, we define a place as a small area that can roughly be represented by a grid in stage one.

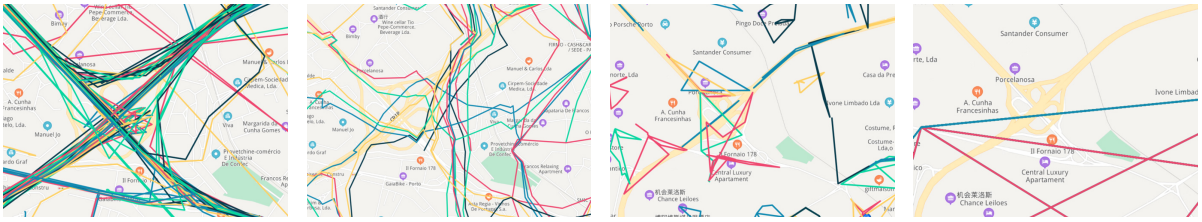
Table 1: Distance between distribution of real and synthetic dataset. We can see that TSG receives the smallest JSD of overall distribution and distance length distribution.

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**



(a) (b)

Figure 7: (a) and (b) show proportion of visits of the top 50 visited places among original dataset and synthetic dataset, which share a similar distribution.



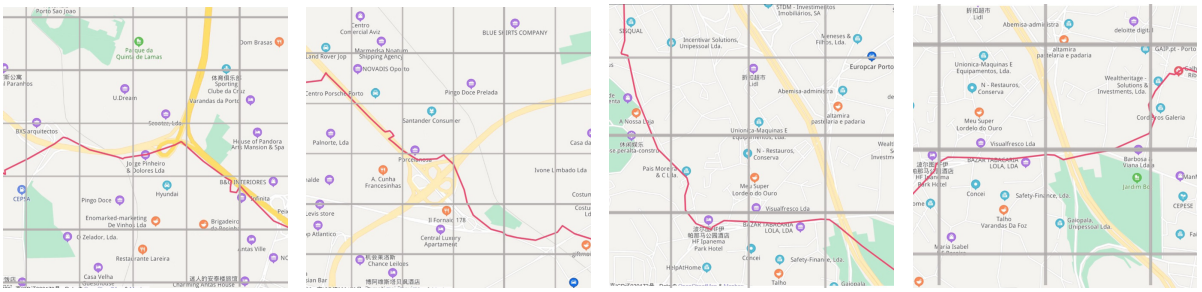
(a) Ground Truth (b) TSG(Ours) (c) FTS-IP (d) LSTM

Figure 8: Part of the synthetic trajectory dataset on the map, compared with ground truth, FTS-IP and LSTM model.

4.4 Road Networks Matching Accuracy

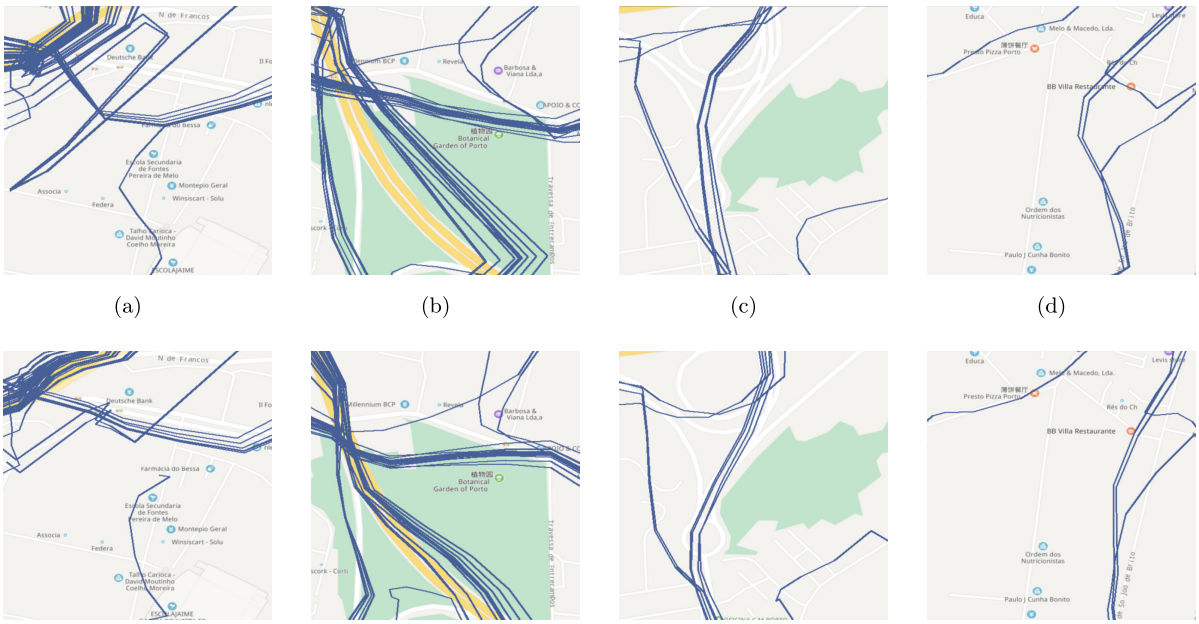
Road network matching accuracy is what makes the synthetic trajectory plausible. Here, we draw our synthetic trajectory on a map to compare its plausibility with other models (see Figure 8 and Figure 9).

Embedding road information into a sequential model by encoding map image with FCN network plays a significant role in ensuring the synthetic trajectories match with corresponding road networks. To validate the effectiveness of our condition in second stage GAN, we compare the road network accuracy with respect to the encoder (see Figure 10). Since the road information



(a) Trajectory a (b) Trajectory b (c) Trajectory c (d) Trajectory d

Figure 9: Examples of our long generated trajectories. They achieve quite satisfying results despite the truth that long trajectories can easily deviate from road networks.



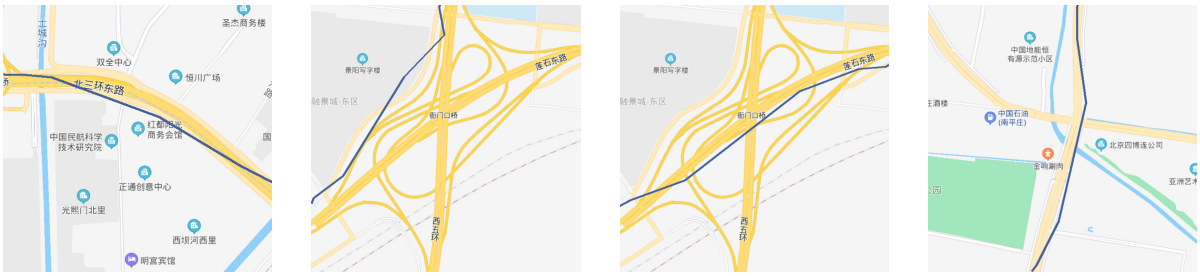
(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 10: Road network accuracy with respect to the encoder. In the first row (a, b, c, d) are results from stage two GAN without the encoder, and each of them has a corresponding picture in the second row (e, f, g, h), showing a better result from stage two GAN with the encoder.

extracted from map images focuses more on main roads and highways, the encoder constrains the synthetic trajectories from deviating from the main roads, as we can tell from the comparison between the result of model with encoder and the one without encoder. However, in terms of big turns and sidewalks (cars could be parked there), our encoder didn't provide such strong constraints.

Also, in order to evaluate the generalization of our model, we test on the map of Beijing, which is not included in our dataset, as shown in Figure 11. The result shows that the model can still generate satisfying trajectories based on the new-coming maps.



(a) (b) (c) (d)

Figure 11: The result on the map of Beijing, which is not included in our dataset. The figures show that the model can still generate trajectories based on the new-coming maps.

5 Conclusion

In this paper, we tackle the problem of generating a large volume of synthetic but realistic trajectories. We consider the pattern and the distribution of the original trajectory dataset and road network information in the corresponding area as two main properties we want our synthetic trajectories to preserve. We propose a novel step-by-step method based on GAN and RNN to learn the joint distribution of the original trajectories and embed road networks from map images respectively. To prevent synthetic trajectory from deviating from the direction we previously set, we also propose a novel network structure that is consisted of two symmetric LSTMs as the decoder in the second stage. The loss function has also been adjusted to a weighted one for putting more constraints on the first several trace points. During experiments, we show the efficacy of our method on both capturing distribution and matching road networks.

However, sometimes the real-world cases are more complex than our assumption. The GPS trajectory may vibrate at a specific point for a long time maybe because he or she went to the supermarket, or the GPS signal is unstable. The traces data in these cases come from a different distribution, and their sequence length will be much longer, which our model cannot predict well without additional information.

In our future work, we will continue to investigate how to generate right tail data. We can try to build models separately for different categories of traces, to consider more detailed features such as driving behaviors. Also we will try to replace random input Z with a different type of distribution, like Gamma distribution rather than the normal distribution to generate more right tail data.

Besides, we will continue to investigate how to adjust the network structure of the decoder in the second stage, which may be able to simplify the linking algorithm that connects the two stages. We will also look into other possible forms of road information than map images.

Supplementary Material

The trajectories data of Porto is available on Kaggle (http://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i). Our Python code in experiment section can be found at https://github.com/XingruiWang/Two-Stage-Gan-in-trajectory-generation.

References

Alahi A, Goel K, Ramanathan V, Robicquet A, Fei-Fei L, Savarese S (2016). Social LSTM: Human trajectory prediction in crowded spaces. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 961–971.

Andreini P, Bonechi S, Bianchini M, Mecocci A, Scarselli F, Sodi A (2019). A two stage gan for high resolution retinal image generation and segmentation. arXiv preprint: https://arxiv.org/abs/1907.12296.

Baratchi M, Meratnia N, Havinga PJM, Skidmore AK, Toxopeus BAKG (2014). A hierarchical hidden semi-Markov model for modeling mobility data. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, UbiComp '14*, (AJ Brush, A Friday, JA Kientz, J Scott, J Song, eds.), 401–412. Association for Computing Machinery, New York, NY, USA.

Barth A, Franke U (2009). Estimating the driving state of oncoming vehicles from a moving platform using stereo vision. *IEEE Transactions on Intelligent Transportation Systems*, 10(4): 560–571.

Bengio S, Vinyals O, Jaitly N, Shazeer N (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In: *Advances in Neural Information Processing Systems*, (CC Cortes, ND Lawrence, DD Lee, M Sugiyama, R Garnett, eds.), 1171–1179.

Chen C, Mu S, Xiao W, Ye Z, Wu L, Ju Q (2019). Improving image captioning with conditional generative adversarial nets. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, (A Cohn, Emeritus, K Ford, eds.), volume 33, 8142–8150.

Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (A Moschitti, B Pang, W Daelemans, eds.), 1724–1734. Association for Computational Linguistics, Doha, Qatar.

Dai J, Yang B, Guo C, Ding Z (2015). Personalized route recommendation using big trajectory data. In: *2015 IEEE 31st International Conference on Data Engineering*, (JG Gehrke, WL Lehner, KS Shim, SK Cha, GM Lohman, eds.), 543–554.

Duan J (2017). Two stage GAN. https://davidsonic.github.io/Project/GAN.html (Accessed on 12/05/2020).

Ferguson D, Darms M, Urmson C, Kolski S (2008). Detection, prediction, and avoidance of dynamic obstacles in urban environments. In: *2008 IEEE Intelligent Vehicles Symposium*, 1149–1154.

Gao Q, Zhou F, Zhang K, Trajcevski G, Luo X, Zhang F (2017). Identifying human mobility via trajectory embeddings. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, (C Sierra, eds.), 1689–1695.

Garg S, Peitz S, Nallasamy U, Paulik M (2019). Jointly learning to align and translate with transformer models. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP)*, (K Inui, J Jiang, V Ng, X Wan, eds.), 4453–4462. Hong Kong, China. November 3–7.

Gehring J, Auli M, Grangier D, Yarats D, Dauphin YN (2017). Convolutional sequence to sequence learning. In: *Proceedings of International Conference on Machine Learning*, (D Precup, Y Whye, eds.), 1243–1252.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. (2014). Generative adversarial nets. In: *Advances in Neural Information Processing Systems*, (Z Ghahramani, M Welling, C Cortes, N Lawrence, KQ Weinberger, eds.), volume 27, 2672–2680. Curran Associates, Inc. 3226489310

Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017). Improved training of Wasserstein GANs. In: *Advances in Neural Information Processing Systems*, (I Guyon, UV Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, R Garnett, eds.), volume 30, 5767–5777. Curran Associates, Inc.

Hochreiter S, Schmidhuber J (1997). Long short-term memory. *Neural Computation*, 9(8): 1735–1780.

Huang Z, Xu W, Yu K (2015). Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint: https://arxiv.org/abs/1508.01991.

Kulkarni V, Garbinato B (2017). Generating synthetic mobility traffic using RNNs. In: *Proceedings of the 1st Workshop on Artificial Intelligence and Deep Learning for Geographic Knowledge Discovery*, (H Mao, Y Hu, B Kar, S Gao, G McKenzie, eds.), 1–4.

Kulkarni V, Tagasovska N, Vatter T, Garbinato B (2018). Generative models for simulating mobility trajectories. In: *Workshop on Modeling and Decision-Making in the Spatiotemporal Domain, 32nd Conference on Neural Information Processing Systems (NIPS 2018)*, Montréal, Canada.

Li J, Chen W, Liu A, Li Z, Zhao L (2018). FTS: A feature-preserving trajectory synthesis model. *Geoinformatica*, 22(1): 49–70.

Liu Y, Kang C, Gao S, Xiao Y, Tian Y (2012). Understanding intra-urban trip patterns from taxi trajectory data. *Journal of Geographical Systems*, 14(4): 463–483.

Long J, Shelhamer E, Darrell T (2015). Fully convolutional networks for semantic segmentation. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431–3440.

Lu J, Xiong C, Parikh D, Socher R (2017). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 375–383.

Mohajerin N, Rohani M (2019). Multi-step prediction of occupancy grid maps with recurrent neural networks. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10592–10600.

Ouyang K, Shokri R, Rosenblum DS, Yang W (2018). A non-parametric generative model for human trajectories. In: *International Joint Conferences on Artificial Intelligence*, (J Lang, eds.), 3812–3817.

Shokri R, Theodorakopoulos G, Le Boudec JY, Hubaux JP (2011). Quantifying location privacy. In: *IEEE Symposium on Security and Privacy*, 247–262.

Srikanth S, Ansari JA, Ram RK, Sharma S, Murthy JK, Krishna KM (2019). INFER: INtermediate Representations for FuturE pRediction. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 942–949.

Vinyals O, Toshev A, Bengio S, Erhan D (2015). Show and tell: A neural image caption generator. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3156–3164.

Wang Z, Lu M, Yuan X, Zhang J, Van De Wetering H (2013). Visual traffic jam analysis based on trajectory data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12): 2159–2168.

Weikum G (2002). Foundations of statistical natural language processing. *SIGMOD Record*, 31(3): 37–38.

Wu Q, Shen C, Liu L, Dick A, Van Den Hengel A (2016a). What value do explicit high level concepts have in vision to language problems? In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 203–212.

Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, et al. (2016b). Google's Neural Machine Translation system: Bridging the gap between human and machine translation. arXiv preprint: https://arxiv.org/abs/1609.08144.

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, et al. (2015). Show, attend and tell: Neural image caption generation with visual attention. In: *International Conference on Machine Learning*, (FR Bach, DM Blei, eds.), 2048–2057.

Yue Y, Zhuang Y, Li Q, Mao Q (2009). Mining time-dependent attractive areas and movement patterns from taxi trajectory data. In: *2009 17th International Conference on Geoinformatics*, 1–6.