博士論文

Citywide Human Mobility Estimation using Deep Learning
（深層学習を利用した都市スケールでの人々のモビリティ推定）

黄　豆

Abstract

With the progress of urbanization, more and more people are now living in urban areas with very high population density. The agglomeration of population and industries has improved the cities' production efficiency as well as provided many conveniences for the people living there. However, at the same time, dense urban areas have also emerged many new challenges due to the concentration of population and industries. In order to find possible solutions to these challenges, many different types of data sources have been collected from cities via different sensors. We divide the collected data into two kinds: data with the static feature, and data with the dynamic feature, according to what kind of information they contained. For example, road networks, street maps, et al., mainly reflect the static characteristics of the city. In contrast, surveillance videos, GPS positioning data, et al. mainly reflect on the dynamic characteristics of the town. These continuously collected urban data gave birth to the concept of urban computing. In recent years, many studies use these data with the dynamic characteristics of cities to solve different problems in cities, such as traffic forecasting, urban planning, et al. However, we must admit that urban sensing and data acquisition are still a significant challenge. Given the two main problems existing in GPS trajectory data that simulate citywide human mobility: 1) privacy protection, 2) sampling deviation and noise, we hope to use deep learning-based methods for human mobility estimation.

In Chapter 2, we first briefly introduced the concept of generative models and the reasons for using generative models for human mobility estimation. We first define the problem of citywide human mobility estimation as a Bayesian inference problem. We assume that no matter how complex a human mobility trajectory is, we can always find a probability distribution in the hidden space to characterize the information contained in this complex trajectory. Hence, the key to human mobility estimation is finding the posterior distribution of this hypothetical hidden space distribution when the actual trajectory is observed. We hope to solve this problem because the hidden space distribution contains the necessary information for each complex trajectory. At the same time, we can avoid the possibility of privacy violation when using the trajectory directly. Furthermore, suppose we can find the joint distribution of the hidden space distribution and the actually observed trajectory. In that case, we can also sample from the hidden space distribution to reduce the sampling bias of the collected trajectory data.

This chapter uses the framework of variational inference to solve this inference problem and uses LSTM as Encoder and Decoder to complete the conversion between trajectory data and hidden space distribution. The experimental results show that our method can achieve our goal. Nevertheless, at the same time, we also discovered the limitation that the newly generated virtual trajectory does not comply with the constraints of geographic information.

Chapter 3 considered the advantages and limitations of directly using the generative model for human mobility estimation and considered improving two of these limitations. The first limitation is that when we generate new virtual data directly using the generative model, the newly generated trajectory data does not comply with geographic information constraints. That is, the car could appear in an area outside the road network. The second limitation is that it is difficult for us to quantitatively measure the authenticity of the newly generated virtual trajectory data. We naturally think that we can use map matching to match the newly generated trajectory to the road network regarding the first limitation. However, we have noticed that such post-processing will change the citywide human mobility pattern. Therefore, we first used the shortest distance with the map matching method to conduct a simple experiment to measure the change of the citywide human mobility pattern of the post-processing virtual trajectory. The experimental results show that the citywide human mobility pattern represented by the post-processing trajectory data has probably changed by more than 20%. Then, from the perspective of trajectory similarity, we use the retrieval idea to construct a retrieval-based human mobility estimation model. In this way, we can avoid the human mobility pattern change brought by map matching as post-processing and avoid the problem of quantifying the authenticity of the newly generated virtual trajectory. We first use the deep learning model to convert complex trajectories into hidden space distributions. We then use the distance between the hidden space distributions corresponding to different trajectories to complete a quick search with the k-d tree technique. In the experiment, we compared the deep learning model with the traditional trajectory similarity method. We found that the deep learning model, especially based on the two-way LSTM and VAE model, obtained the best results. The limitation of the retrieval-based model is that we need a vast historical trajectory database, and we do not know how to estimate the appropriate weight of each observed trajectory in citywide human mobility.

Chapter 4 proposed a differentiable projection method to construct a deep learning

model with linear constraints. This problem is worth studying because in some scenarios where we can get some simple prior knowledge, constrained deep learning always gives better results than conventional deep learning models as a pure data-driven algorithm. In this chapter, we give the theoretical derivation as a piece of evidence for the above conclusions. In our understanding, prior knowledge and constraints are synonymous, so that we can treat the information provided by some heterogeneous data as constraints. This chapter provides a theoretical basis and implementation method for the work of the next chapter 5. As long as the information provided can be written in a linear form, we can use the method of this chapter to model. We start from a linear equality constraint conditions problem. We could give a projection method for solving this linear equality constrained problem if the constrained conditions are independent. However, there is no straightforward way to use the projection method to solve the linear inequality constrained problems. However, fortunately, we can use a partial projection algorithm to make the projected points closer to the constrained region than the original points. Then we propose a differentiable projection method for deep learning based on this theorem. At the same time, we also used some synthetic data to conduct experiments to verify the effectiveness of this method.

In Chapter 5, we made a simple application of the constrained network model proposed in Chapter 4 in human mobility estimation. The framework proposed in this chapter is an improvement to the limitations of Chapter 3. We hope to estimate the individual human mobility trajectory while also estimating the weight of this trajectory in citywide human mobility. To this end, we used simulated heterogeneous OD data to construct our constraint information. First, we convert the trajectory data into a Gaussian mixture of hidden space distribution. Then our Encoder also needs to output the weight of this trajectory to construct citywide human mobility. Since we use constrained deep learning as our Encoder, the weight of each output can satisfy the constraint information, which makes our results better than the conventional deep learning model. The experimental results show that the proposed constrained human trajectory generation model has the best results in estimating citywide human mobility without losing the ability to express individual trajectories.

In Chapter 6, we show the work we have done so far, summarize our contributions, and discuss the limitations of the current work. In the future, a possible direction is to carry out a systematic integration of all current work. Besides, it is worth trying to use more kinds of data under the current framework for citywide human mobility estimation.

At the same time, we believe that data fusion is also the direction that this research can extend in the future. Because through our current work, we find that the current method has proposed a way to solve privacy infringement when using human mobility data directly. However, a limitation lies in that although our current work reduces the problem of data sampling bias to a certain extent, it is still limited to the use of a single data source, so the final performance still has much room for improvement. In the future, we think we still need to mine more practical information from more different data sets by fusing different data sources to complete human mobility estimation with minor sampling bias.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With urbanization, many people spend their whole lives in some highly developed dense urban areas, sometimes called megacities, such as Tokyo, New York, and Shanghai. Some statistical evidence shows that the efficiency of energy consumption in service establishments is higher than in sparse rural areas in these megacities. More precisely, research from Morikawa [1] said that energy efficiency increase by approximately 12% when the density in a municipality population doubles after controlling for differences among industries quantitatively. On the one hand, those big cities create possibilities for people to have a modernized lifestyle. On the other hand, it also engendered many challenges, such as air pollution, increased energy consumption, and traffic congestion, of course. Taking traffic congestion in big cities as an example, people waste their time in traffic congestion when commuting to work in rush hours. It reduces the quality of life of every individual and limits the further development of the city. Therefore, much research focuses on tackling those challenges in multiple city-related fields, such as transportation, civil engineering, environment, economy et al.

Collecting information from the urban area, in reality, is the basis for tackling those challenges mentioned above. Research in different city-related fields focuses on some specific information of the urban area. For example, transportation research may focus on traffic congestion information in the road network and then advise city planning to prevent congestion in rush hours. Here we give our understanding of essential information in the context of urban space. We use features to denote the essential information

of cities, and then the features contain static features and dynamic features. The static features like the road network, Point of Interest (POI) change slowly in the long term. Then, the dynamic features are mainly activities generated by people's mobility, such as walking in streets, driving on roads, and taking trains. The static features and the dynamic features interact with each other. For example, people may spend more time on a trip in the insufficient road network, and reduce the efficiency of the entire city.

Collecting information on static features of cities is much easier than collecting dynamic features even now. Since the static features of cities change slowly, we can draw some maps for cities and record the road network and POIs. In recent years, with the advancement of Remote Sensing (RS) techniques, more detailed and timeliness maps can be drawn for extracting that static information of cities. Therefore, in many early works of city planning, tackling challenges of the urban area is highly dependent on static information. At that time, policymakers in rapidly developing cities often make decisions about city planning using developed cities as a reference to prevent some issues already exist before. They use the experience of those developed cities to build their developing cities, while this is not an easy task. We give an example of the strategy of improving the traffic efficiency in some cities. In some developing cities, vast roads are built even though the number of vehicles is not significant. This kind of planning reduces the possibility of traffic congestion in the future. In contrast, it reduces the accessibility of pedestrians because people may take more than 10 minutes to cross a street. Therefore, the city is built to be convenient to drive, not to live. Besides, the improvement can hardly be made before we get an effective solution for those issues in cities since the cost of infrastructure improvement is significantly huge for those cities already developed. Therefore, we need more factual information about human mobility, which is a dominant factor that forms the dynamic features of cities. Fortunately, a wide variety of data is generated in urban spaces, and we can capture these data with the development of sensing technologies in recent years. In the field of human behavior tracking, there are active sensors like Radio-frequency identification (RFID) tags, WIFI, Bluetooth sensors on mobile devices, and passive sensors like cameras, passive infrared (PIR) motion detectors [2]. Those sensing techniques make the information acquisition of urban spaces possible and provide opportunities for building more intelligent cities.

Much research about applications tackling challenges in big cities based on this kind of Spatio-temporal information has been conducted in recent years. Zheng [3] gives a more specific definition of urban computing, including data acquisition, integration,

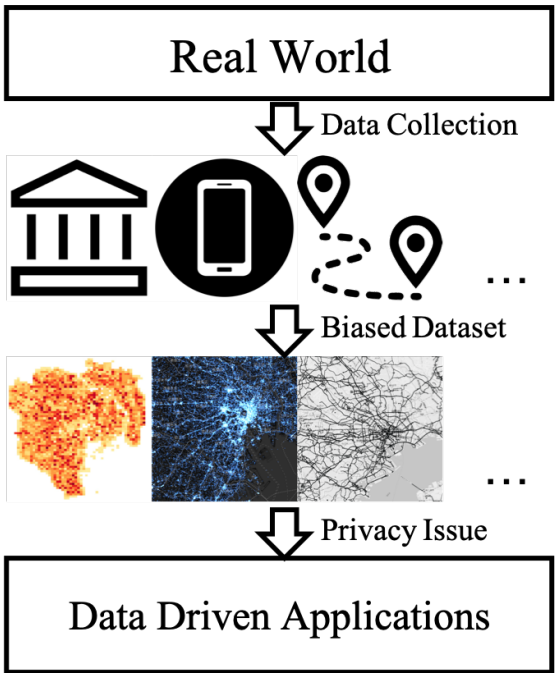


FIGURE 1.1: Data-driven applications rely on information in datasets collected from the real world. However, collected datasets cannot reflect the real world. These datasets are sometimes unavailable because the collected datasets are often surfer from sampling bias and privacy violation problems.

and extensive and heterogeneous data analysis in urban spaces to tackle issues in cities. Although there is already extensive research conducted based on existing data sources, urban sensing and data acquisition are still among the main challenges in urban computing. It is a nontrivial issue to collect citywide data unobtrusively and continually since we do not have enough sensors in every road segment or buildings. A direct counter method is installing more sensors among cities, but this will dramatically increase the cost of the infrastructure of cities. A new concept such as humans as sensors gives us a more flexible solution to leverage what we already have in urban space more intelligently. For example, human mobility in a city that occurs during rare events like an earthquake was recorded. We could use this data to evaluate the situation if the earthquake happened in another city [4]. However, it still has two main challenges:

1) Privacy violation issue: There are four main approaches to protecting users from privacy violations caused by spatial-temporal data acquisition [5]. The intuition behind those approaches is reducing the representative power of data by suppressing the location information of users. It means that even though enterprises or the government have collected high-quality human mobility data, it cannot be used directly sometimes.

2) Bias and noise: Nonuniformly distributed sensors cause bias of collected human mobility data. There may be no people at some moments in some places, which inevitably leads to data sparsity problem, while in some other places, like Tokyo station, the data generated by users may be redundant. Besides, the human mobility data generated by users may be very noisy, unlike data collected by some traditional sensors. To prevent

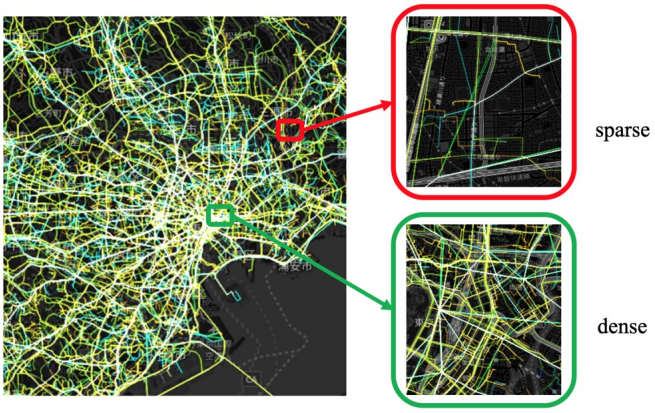


FIGURE 1.2: Visualization of human mobility trajectories in Tokyo to show the bias problem.

some risks of violation of the privacy of mobile device users, although many data is collected, the owner of the such kind of data is not willing to provide the data for researchers or other research institutes, which leads to a limitation of the usage of this kind of locational data. Despite the privacy concerns about protecting users' privacy, there is a bias problem caused by the low sampling rate of the collected data. Both privacy concerns of mobile device users and the data bias issue will lead to the difficulty to obtain the human mobility data to reflect the actual trajectory patterns in real situations. Therefore, we want to build an urban trajectory estimator that can convert the privacy-sensitive real trajectory data to an estimated data to avoid privacy violation. This estimated data should have a better representation of the real world.

In recent years, deep learning contributes to solving issues that have been resisted the best attempts of the Artificial Intelligence community for many years. Generally speaking, deep learning is a kind of method which usually stacks multiple simple nonlinear transformation module as a kind of complex and multi-layer representation learning method [6]. The key idea is that when given data with potentially complex representation, a deep learning model captures data representation at one level at each stacked simple transformation module, which gradually forms a more abstract representation at the top layer of the deep learning model. The critical aspect of the deep learning method,

comparing with the conventional Machine Learning method, is that it can learn features from data automatically using a general-purpose procedure which requires less feature engineering and expertise of a specific domain. In recent years, it shows state-of-the-art performance in many fields, such as computer vision, medical applications, and natural language processing.

Furthermore, many deep learning models such as Recurrent Neural Networks (RNNs) continue to mature. New deep learning-based methods are constantly being proposed to solve various problems that rely on time-series data, such as weather forecasts. These deep learning models can fully mine the critical features in the data without the need for additional expert knowledge in many cases and have obtained state-of-the-art performance. Therefore, in our research, we use deep learning methods to model human mobility trajectories.

1.2 Related work

Research on huam mobility model Subsequently, pure random mobility models occupied this field of research for many years until a turning point appeared in 2005. Barabsi [7] studied various human activities, including email communication, online chat, etc. Many human behaviors systematically deviate from the Poisson process, mimicking the random distribution of human behavior over time. This groundbreaking work demonstrated the sudden non-Poisson nature of human activities and inspired many researchers to explore human dynamics' intrinsic nature further. Brockmann et al. [8] analyze the recorded data of banknotes in the United States. It is found that the dispersion of people follows a power law, which provides a basis for quantitative analysis of people's movement characteristics. After that, Gonzalez et al. [9] analyzed individual human mobility patterns based on the trajectories of 100,000 anonymous mobile phone users for six months. The results show that even though the historical trajectories of individuals are highly diverse, they do not satisfy the random prediction made with levy flight or random walk but tend to be a more reproducible and straightforward pattern. This work provides a specific basis for the reconstruction of the human mobility model. Their results showed firm shreds of evidence of the periodicity, boundedness, and regularity of human mobility. Initially, random models such as Random Direction (RD), Random Way Point (RWP), and Random Walk (RW) were widely used to construct

human mobility models. Some literature [10] referred to these models as Pure Random Mobility Model", which synthesizes human mobility traces through some pre-defined specific probability distributions, which actual observations have not verified. For example, RWP [11] is a representative pure random mobility model because this model integrates human movement simplified into a line segment with pauses. These line segments are independently randomly sampled and synthesized. Rhee et al. [12] mentioned that some previous human mobility models, such as random way point (RWP), did not consider the tendency of people's daily movement. They used a truncated Levy walk mobility (TLW) model to illustrate that heavy-tail features can be an essential component incorporated in human mobility models to improve the authenticity of human mobility generated by the model. At the same time, Lee et al. [13] improved the previous human mobility model based on the statistical characteristics of human mobility proposed by previous studies. Then they considered two more statistical features, the destination of people and the tendency of a destination to be close to the current point, and proposed a method called Self-similar Least-Action Walk (SLAW) to synthesize human mobility traces artificially. Ghosh et al. [14] proposed Sociological Orbit aware Location Approximation and Routing (SOLAR) as a feature of human mobility to synthesize human mobility traces in their research. Morlot [15] analyzed user clumps and hotspots during the large-scale urban mass gathering in mobile networks and proposed a new interaction-based mobility model to better describe user clumps' dynamic characteristics. Isaacman [16] proposed and verified "Work and Home Extracted REgions" (WHERE), a regional scale human mobility model. They first identify the key attributes of human movements, such as important locations and commuting distance, and then use the probability distributions obtained from these critical attributes to generate synthetic Call Details Records (CDRs). Calabrese [17] proposed a new human mobility model to predict the position of individuals changing over time. This proposed model has based on the individual's past historical trajectory and geographic information characteristics, such as the collective human mobility pattern, land use, Point of Interest, and the distance of the trip. Jardosh [18] proposed an Obstacle Mobility Model to simulate human movement patterns with the topology of the real world. The model they proposed allows the real-world terrain to be taken into account, which is not available in the previous studies, so their method offers a more realistic human mobility model. Bai et al. [19] proposed various independent indicators to measure whether different human mobility models have captured important information, such as spatial and temporal information

and geographic constraints. They used these indicators to compare various human mobility models and compare different models' performance in different scenarios. Social context information mainly refers to the social relationships between groups of people. The models in [20] and [21] simulate the social environment through an interaction matrix, which quantifies the degree of attractiveness between individuals. Several famous people attract more people to move to them. Then the matrix is used to calculate the transformation pattern of people, which determines the possibility of a person moving between different positions. The periodic movement model [22] simulates the frequent movement of human movement between a set of predefined potential states (locations), such as "home" and "workplace," through independent Gaussian distribution.

Urban computing using human mobility data Techniques of data collection have been improved rapidly, which lead to some revolutionary ideas of implementing machine learning algorithms for solving some traditional social issues, such as zone regulation[23], air pollution[24], disaster evacuation[25] el at., since collected big and heterogeneous data makes tasks which are nearly impossible years ago become possible. Recently, many research pieces have been conducted on human mobility data, such as mobile phone GPS log data, taxi GPS data, and navigation GPS data. These kinds of researches are often related to building an intelligent city system. R, Jiang[26] introduce a framework of predicting multiple steps of future trajectories of human mobility. Their method is a Regions-of-Interest (ROIs) based modeling, which is convinced to be an improvement of traditional grid-based modeling. Also, they use multiple to multiple training strategies to predict the various steps of future movement. CityMomentum[27] is another work related to human mobility prediction. However, building the CityMomentum system is not to predict future trajectories of human mobility using previous historical trajectories but to transfer the information obtained to another city. It is also an exciting work that answers how to use data collected in a town to guide the development of another city. Detecting flawed urban planning using the GPS trajectories of taxicabs[28] is one of the most significant examples of urban computing for city planning. Their work can detect the regions with salient traffic problems and the linking structure and correlation among them. Furthermore, some other researches about simulating human mobility when disasters occur and predict their mobility in an emergency have also

been conducted[4, 29–32]. Their works are significant since understanding and modeling people's mobility is crucial for transportation planning and management.

Researches based on Variational Autoencoder In recent years, Variational Autoencoders (VAEs) have been widely used to approximate some complicated distributions [33]. The ability of VAEs has been proved to be promising in the works of generating many kinds of complex data in the image processing domain. However, some researchers also use this framework in other fields such as Natural Language Processing (NLP), which inspired its implementation to tackle issues based on sequential data. Y. Fan el at.[34] presents a novel end-to-end partially supervised deep learning approach for video anomaly detection and localization using standard samples. It is the first time that A Variational Autoencoder (VAE) framework utilized for video anomaly detection. Y. Pu et at.[35] developed a novel Variational Autoencoder to model images, as well as associated labels or captions. They use a deep Convolutional Neural Network (CNN) as an image encoder, while A Deep Generative Deconvolutional Network (DGDN) is used as a decoder of the latent features. The proposed model achieves high performance on image recognition. Besides, there are many pieces of research using Variational Autoencoders in Natural Language Processing (NLP). Jonas Muller el at.[36] implemented the Variational Autoencoder framework for revising natural language sentences. Comparison between Variational Autoencoder and Encoder-Decoder models for short conversation is made by Shin Asakawa and Takashi Ogata.[37] Another aspect of research based on Variational Autoencoder is improving the VAE framework itself. Sønderby, Casper Kaae, et al. proposed a Ladder Variational Autoencoders[38] which can recursively correct the generative distribution by a data-dependent approximate likelihood. Their model can learn a deeper hierarchy of latent variables than other generative models based on Variational Autoencoder. Research about Infinite Variational Autoencoder is done recently.[39] They use a mixture model where the mixing coefficients are modeled by a Dirichlet process, allowing to integrate over the coefficients when performing inference. Their work shows the flexibility for the applications which have only a small number of available training samples.

Generative model for human mobility simulation Various generative models have been developed To simulate human behavior and moving patterns in recent years.

Input-Output Hidden Markov Model (IO-HMM)[40] was proposed to enable activity-based travel demand models which can protect the privacy of mobile phone users while using this cellular data to simulate synthetic agent travel patterns. Their model achieves a reasonable accuracy when conducting an agent-based microscopic traffic simulation. However, the limitation of the proposed model is that if travel patterns vary significantly over the region, a single model will not capture all areas with a good performance. A Gibbs sampling-based multiple hidden Markov model (GSMHMM)[4], designed as a part of the city-coupling algorithm, can generate simulated trajectories in a city-wide scale area such as Tokyo or Osaka. However, as the model is based on Gibbs Sampling, it needs critical prior knowledge for the GSMHMM to generate new human mobility trajectories. However, HMMs cannot wholly model the temporal dependency of states. To improve the HMMs, Baratchi et al.[41] proposed the Hidden Semi-Markov Model(HSMM), which including the duration of the state into the hidden variables. In general, their works are all based on Hidden Markov Model and focus on reconstructing the trajectories of human mobility following specific probability distribution. Very recently, a non-Parametric generative model for human trajectories has been proposed.[42] They use Generative Adversarial Network (GAN) to produce data points after a simple and intuitive yet effective embedding for location traces designed. It is the first time that deep learning methods implemented in building a generative model for human mobility in our knowledge.

1.3 Research objective

There is a trade-off between the implementation of collected locational data and the concerns about violation of users' privacy to prevent some risks of privacy violation of mobile device users. Despite the privacy concerns about protecting users' privacy, some other problems can also lead to the low sampling rate of the collected data. Both privacy concerns of mobile device users and the lack of techniques of a collection method in some cases will lead to the difficulty to get the human mobility data to reflect the actual trajectory patterns in real situations. There is much research and implementation based on the human mobility data, for instance, human mobility prediction. These applications usually need to use previous steps of trajectories to predict human mobility in the future. We are not talking about the accuracy or performance of such methods in this research. Instead, we are concerned that if we cannot get the human mobility

data that can reflect human mobility patterns in an area, it is difficult to predict future human mobility correctly.

It is different from previous research that we try to find a global latent estimation for representing each trajectory using the generative model. In this case, we will improve the diversity of generated trajectories by sampling more similar trajectories from the estimated latent space. Also, we realize the importance of incorporating information from other heterogeneous data sources for this task.

We summarize the novelty of this research as:

1) Find an estimated latent distribution in vector space to represent each urban human trajectory using a generative model;

2) We propose a retrieval-based trajectory generator for trajectory dataset recovery problem;

3) We propose a differentiable projection module in conventional deep learning models to solve the problems with equality and inequality constraints;

4) We propose a constrained trajectory generator to reduce the small trajectory dataset's bias by building constraints using other heterogeneous data sources.

1.4 Structure of the Thesis

The main body of this thesis is composed of the following five chapters:

Chapter 2: Generative model for human mobility introduces the theoretical concept for finding an estimated latent distribution to represent each privacy-sensitive human mobility trajectory by generative model. Then we analyze the advantages and disadvantages of generating virtual human mobility trajectories using a deep generative model directly.

Chapter 3: Retrieval-based Human mobility generation introduces the idea of using the trajectory similarity concept to generate more virtual human mobility trajectories to avoid hard defining the authenticity of virtual human mobility trajectories.

Chapter 4: Constrained deep learning model introduces a differentiable projection module in conventional deep learning models for solving problems with equality

and inequality linear constraints. This chapter gives some theoretical basis for building linear constrained human mobility generation method in chapter 5.

Chapter 5: Linear constrained human mobility generation gives a specific example of using constrained deep learning models for human mobility generation with the guide information in heterogeneous demographic data. This method can estimate the latent distribution of individual human mobility trajectory and its scaling factor citywide simultaneously.

Chapter 6: conclusion concludes the research as well as its contributions and discusses some possible directions for future research.

Chapter 2

Generative model for human mobility

2.1 Introduction

2.1.1 Background

Many big cities have grown thanks to the rapid urbanization progress, which have modernized many people's lives but also engendered significant challenges [3]. Years ago, solving this kind of challenge seems impossible because of the complex and dynamic settings of cities. Nowadays, some impressive methods of locational datasets collection have shown an opportunity for human mobility applications. For example, human mobility in a city that occurs during some rare events like earthquakes was recorded. How can we use this data to evaluate the situation if the earthquake happened in another city? Although the usage of those kinds of datasets, which owned by enterprises or government, can give us opportunities for some potential applications, they have some limitations two-fold: 1) it has the risk of privacy violation in some cases if used directly; 2) it will contain some bias, or the sampling rate is low.

For some human mobility prediction problem which aims to predict the human mobility in a target area, it is necessary to know the actual situation about the current human mobility. However, in reality, the provided data cannot reflect that actual situation if the data only contains 1% of the entire population in the real world. To tackle this kind of problem, we can develop a scaling factor for each trajectory sample by combining

some information, such as population density, from other data set.

The scaling factor can add more trajectories based on observed trajectories to approximate the actual situation of human mobility in a target area. However, its limitations are also apparent. It can only add some trajectories based on the existed observations. Thus, it is a lack of diversity as different people are assumed to behave somehow differently even though they may be in a similar situation. It is more reasonable to achieve a diversity of trajectories when reconstructing the actual human mobility patterns.

In general, a generative model is a model of the conditional probability of the observable X, given a target y, symbolically, $P(X|Y = y)$.[43] It can be used to generate random outcomes, either of an observation and target (x, y), or of an observation x given a target value y. A generative model is not designed for transportation planning and applications directly. However, we can use this kind of model to improve the existing datasets to match the implementation of other applications. This kind of model can solve the limitations of the aforementioned scaling factor method.

First of all, a generative model can learn a low dimensional feature space that can infer the travelers' pattern from the complex redundant collected locational datasets. Then, we can utilize the learned feature space for transportation planning and applications. Furthermore, if necessary, we can resample from the learned low dimensional feature space to generate a fake dataset with a similar pattern to the real dataset for further use. There are two reasons for generating fake datasets:

1) using generated fake datasets can avoid the risk of violation of customers' privacy;

2) obtain enough data samples if the dataset is too small to be used.

Therefore, the problem of how to build a generative model that can capture the features from accurate human mobility trajectories is a fascinating topic. Nowadays, many deep learning methods have been investigated on image processing, natural language processing, and human mobility prediction, et al. based on virtual neural networks. Many different neural networks have been proposed to solve different problems in various domains. Although different deep learning frameworks were proposed for solving the problems lying on very different implementations, we can still be inspired by those deep learning frameworks. Variational Autoencoders (VAE)[44] are proposed initially for image processing, and many applications using variational autoencoders achieved an excellent performance. However, owing to the structure of variational autoencoders, it can only be implemented in applications using non-sequential data.

To tackle problems of human mobility, which is a kind of sequential data, we need to

use Recurrent Neural Networks (RNN) to build our model. Since vanilla RNNs have difficulties with long length sequence training due to vanishing gradient problems, Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) have been designed and widely used when coming to the long length sequence problem.

2.1.2 Problem definition

We were given a citywide human mobility trajectory data set, denoted as x, which contains complex multimodal information. That dynamic information about human mobility patterns is usually privacy-sensitive, so we want to find its corresponding latent space, denoted as z. The problem becomes to find the posterior distribution of $p(z|x)$, which can be calculated using Bayesian inference like:

$$p(z|x) = \frac{p(z,x)}{\int p(z,x)dz} \tag{2.1}$$

The figure 2.1 illustrates the intuition about using the deep generative model to solve the above problem. We are able to build the citywide human mobility trajectory estimator if we can solve the equation. Moreover, by solving the above equation 2.1, we can know 1) the prior distribution of latent distribution of citywide human mobility $p(z)$, 2) the posterior distribution of latent distribution given observed human mobility $p(z|x)$, 3) the posterior distribution of human mobility $p(x|z)$, which is calculated by

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} \tag{2.2}$$

Then, if we sample some noise vetor $z^\star$ in latent distribution, we can recover the human mobility trajectory $x^\star$.

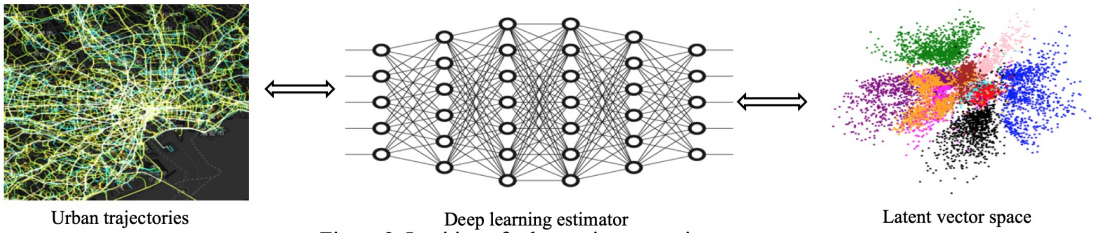


FIGURE 2.1: Intuition of citywide human mobility estimation using generative model.

2.1.3 Research objective

We aim to use a deep generative model to tackle this chapter's citywide human mobility generation problem. In the model architecture, we focus on learning the hidden space of the posterior distribution of human mobility. Then we can resample from the learned distribution and reconstruct the human mobility trajectories. Our objective is as follows:
1) Controlled generation. We want our proposed model architecture to allow us to generate more trajectories that follow the distribution of the learned human mobility pattern. Moreover, we can control the number of trajectories we want to generate.
2) Diversity. We can obtain some reasonable virtual human mobility trajectories, which are not contained in the original data set, to achieve diversity.
3) We will use some metrics to quantitatively evaluate the performance of the proposed model for trajectories of human mobility.
We use real navigation GPS data to experiment. The data we used is locational data contains trajectories of human mobility of the entire Japan.

2.2 Methodology

2.2.1 Preliminary

Variational inference We will use a variational inference method to solve Eq. 2.1 since its denominator is usually intractable due to the integral. Following the variational inference method, instead of directly find the intractable posterior distribution $p(z|x)$, we select some approximate distribution, denoted by $q(z)$, from a family of variational distributions. To measure the gap between selected approximate distribution and the actual intractable posterior distribution, we use Kullback-Leibler Divergence between these two distributions:

$$D_{KL}(q||p) = \int q(z) \log \frac{q(z)}{p(z|x)} dz \tag{2.3}$$

By minimizing the KL-divergence, we could find an approximate close distribution of the posterior distribution. In most cases, minimizing Eq. 2.3 is difficult, since the KL-divergence contains the unknown posterior distribution $p(z|x)$. We rewrite the above

equation as:

$$\begin{aligned} D_{KL}(q||p) &= E_q[\log \frac{q(z)}{p(z|x)}] \\ &= E_q[\log q(z)] - E_q[\log p(z|x)] \\ &= E_q[\log q(z)] - E_q[\log p(x,z)] + \log p(x) \end{aligned} \tag{2.4}$$

Given an observation dataset, its logarithm of distribution $\log p(x)$ should be a constant. Therefore, we define an Evidence Lower Bound (ELBO) by:

$$ELBO(q) = -E_q[\log q(z)] + E_q[\log p(x,z)] \tag{2.5}$$

The ELBO is defined according to following process:

$$\begin{aligned} \log p(x) &= \log \int p(x,z)dz \\ &= \log \int \frac{p(x,z)q(z)}{q(z)}dz \\ &= \log E_q[\frac{p(x,z)}{q(z)}] \leq E_q[\log \frac{p(x,z)}{q(z)}] \\ &= E_q[\log p(x,z)] - E_q[\log q(z)] \end{aligned} \tag{2.6}$$

We use Jensen's inequality on the log probability of the observations here. Then, minimizing KL-divergence is equivalent to maximizing ELBO, which means that we are able to find an approximate posterior distribution.

Variational Autoencoder Autoencoders is widely used for the generation before. However, its fundamental problem is that the latent space, constructed by the Autoencoder from learning the features of input data, may not be continuous or allow easy interpolation. The purpose for building a generative model is that we want to randomly sample more data from the approximate latent space or generate variations on input data from a continuous latent space. When the latent space constructed has discontinuities, the decoder will simply generate an unrealistic output if we sample or generate a variation from there. That is because the decoder cannot deal with that region of the latent space since it never saw such an encoded vector from that region of latent space during training. One fundamentally unique property of Variational Autoencoders (VAEs), which separate them from vanilla Autoencoders, is that their latent space is designed to be continuous, allowing easy random sampling and interpolation. It is also this property

that makes Variational Autoencoders useful for generative modeling. That property is achieved by making its encoder output two vectors of size n: a vector of means μ, and another vector of standard deviation σ, instead of just output one single encoding vector of size n. These two encoding vectors then form the parameters of a vector of random variables of length n, with the i-th element of μ and σ being the mean and standard deviation of the i-th random variable X_i, from which we sample to obtain the sampled encoding which we pass onward to the decoder. This stochastic generation means that even for the same input. In contrast, the mean and standard deviations remain the same. The actual encoding will somewhat vary on every single pass simply due to sampling.

Intuitively, the main difference between the constructed latent spaces of a standard Autoencoder and a Variational Autoencoder. In the latent space of a Variational Autoencoder, the encoded mean vector μ and the standard deviation σ initialize a probability distribution. In contrast, the encoded vector of a standard Autoencoder is a direct encoding coordinate. In the case of training a Variational Autoencoder, encodings can be generated randomly from the probability distribution. Therefore, the decoder of a Variational Autoencoder can learn to reconstruct the output from a probability distribution rather than just a group of specific points in the latent space. Kullback-Leibler divergence[45] is a measure of how one probability distribution diverges from a second, expected probability distribution. The most critical metric in information theory is Entropy which is to quantify the information in data. The definition of Entropy for a probability distribution $p(x)$ is:

$$H = -\sum_{i=1}^{N} p(x_i) log p(x_i) \tag{2.7}$$

Based on the formula of entropy, the Kullback-Leibler divergence which measures the difference between a probability distribution $p(x)$ and the approximating distribution $q(x)$ can be given:

$$D_K L(p||q) = \sum_{i=1}^{N} p(x_i)(log p(x_i) - log q(x_i)) \tag{2.8}$$

With Kullback-Leibler divergence, we can calculate precisely how much information is lost when we approximate one probability distribution with another one. The encoder of a Variational Autoencoder is designed to convert the input data point x to a hidden

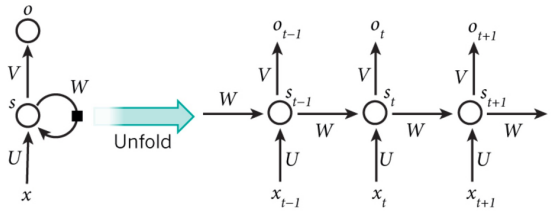


FIGURE 2.2: architecture of Recurrent Neural Network.

representation z, with weights and biases θ. Therefore, the encoder is denoted to be $q_\theta(z|x)$. The noisy values of hidden representation z are sampled from this distribution as the input of the decoder. The decoder of a Variational Autoencoder has weights and biases ϕ, denoted by $p_\phi(x|z)$. It gets the noisy values of the latent representation z as input and reconstructs the output data x. The reconstruction log-likelihood $logp_\phi(x|z)$ is used to measure the information lost in the procedure mentioned above. It also gives the efficiency of the decoder for reconstructing input data x given its latent representation z. The loss function of the Variational Autoencoder is:

$$l_i(\theta, \phi) = -E_{z\sim q_\theta(z|x_i)}[logp_\phi(x_i|z)] + KL(q_\theta(z|x_i)||p(z)) \tag{2.9}$$

It contains two-part: 1) the first term is named to be reconstruction loss; 2) the second term is a Kullback-Leibler divergence between the probability distribution of encoder and a unit Gaussian distribution. This loss function is well designed as we can also treat the second term to be a regularizer, just like many other loss functions. A reconstruction loss forces the model to give the output just as similar as possible compared with input. Meanwhile, the purpose of the second term is to make sure the latent space constructed in the training process is not complex. When the second term is small, we can use a simple latent space to approximate the real posterior distribution of latent space.

Long-Short Term Memory Recurrent Neural Networks (RNN) are widely used in solving many sequential problems such as Natural Language Processing (NLP) tasks.[46–48] The main contribution of RNNs is that they can capture sequential information for use. For instance, it is a good idea to obtain the previous location, which a data point located in before we predict the next location where the data point will be. A typical RNN is shown as figure 2.2. Input data is denoted by $x = (x_1, ..., x_{t-1}, x_t, x_{t+1}, ...)$. An observation x_t indicates the observed data in step t. Corresponding to the input data, a hidden space is denoted by $s = (s_1, ..., s_{t-1}, s_t, s_{t+1}, ...)$. However, a hidden space

is not only related with input data but also related with previous hidden state: $s_t = f(Ux_t + Ws_{t-1})$. The functin $f(\cdot)$ is nonlinear activation such as ReLU or tanh. A hidden state s_t can capture the information of current observation x_t and take the previous information captured by s_{t-1} into account. Then, the output $o = (o_1, ..., o_{t-1}, o_t, o_{t+1}, ...)$ can be calculated by $o_t = g(Vs_t)$, where function $g(\cdot)$ is another nonlinear activation. From figure 2.2, we notice that weights U, V, and W only be changed after a sequence be computed completely. Therefore, the total numbder of parameters of a Recurrent Neural Network is not so big comparing with other traditional deep neural networks. However, there is a main difference between Recurrent Neural Network and some other traditional deep neural networks, which is that the backpropagation algorithm is used for training a traditional neural networks while it cannot be used for training a Recurrent Neural Network. That is because the gradient at each output depends on not only in current step, but also previous steps. In that case, a specific backpropagation is designed for traing a Recurrent Neural Network which is called Backpropagation Through Time (BPTT). Long-Short-Term Memory (LSTM) was designed to combat vanishing gradients through a gating mechanism [49]. How a LSTM calculates a hidden state s_t is shown as follows:

$$\begin{aligned} i &= \sigma(x_t U^i + s_{t-1} W^i) \\ f &= \sigma(x_t U^f + s_{t-1} W^f) \\ o &= \sigma(x_t U^o + s_{t-1} W^o) \\ g &= \tanh(x_t U^g + s_{t-1} W^g) \\ c_t &= c_{t-1} \circ f + g \circ f \\ s_t &= \tanh(c_t) \circ o \end{aligned} \tag{2.10}$$

A LSTM layer, shown in figure 2.3, has three gates i, f, o. i is called input gate, f is output gate, and o is output gate. The sigmoid function is used in thse gates which has values between 0 and 1. For example, if the value of a gate is 1, then it means that let all information pass towards, while if the value of a gate is 0, it means that no information shall passed onwards. The function of different gates is different. The input gate i determines the quantity of information of current input to be passed onwards. The forget gate f determines the quantity of information of previous state to be passed onwards. The output gate o determines the quality of information of internal state to be passed onwards. Besides, g is designed to be a kind of candidate hidden state which is

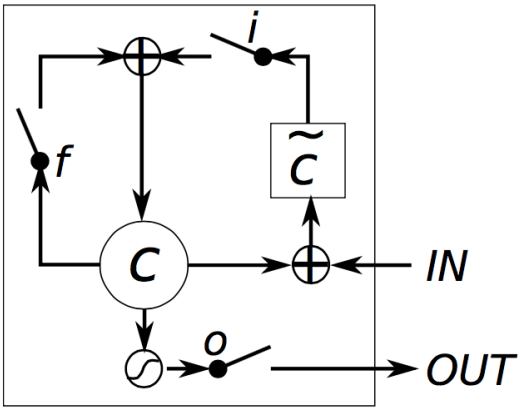


FIGURE 2.3: architecture of Long-Short-Term Memory (LSTM).

also calculated based on the current input x_t and the previous output s_{t-1} just like the hidden state calculated in a vanill Recurrent Neural Network. However, This candidate hidden state is not the final hidden state calculated in a Long-Short-Term Memory as it should be selected by the aforementioned input gate. c_t in a Long-short-Term Memory unit is the internal memory. It is used to capture the information combining the previous internal memory c_{t-1} with selected candidate hidden state g. Using a internal memory, we can completely ignore the previous memory by set the value of the forget gate to be 0, or completely ignore the new input by set the value of input gate to be 0. However, what we really want is information between these two extremes. Finally, we can compute the output hidden state s_t using the internal memory c_t. Since there is a output gate which control the quantity of information to be passed onwards, the hidden state s_t could contain only part information of the internal hidden state. The ability of modeling long-term dependencies is improved in LSTMs thanks to the gating mechanism.

2.2.2 Framework

Since urban trajectory data is a kind of sequence data, using Long Short-Term Memory (LSTM) networks is a natural choice. LSTM network is a kind of improved Recurrent Neural Network (RNN) which uses a gated mechanism designed to solve the vanishing gradient problems happened in vanilla RNN networks. Here, we use two LSTMs to handle the urban trajectory dataset. One LSTM is used as encoder which take trajectory data as input then gives compressed low dimensional vectors as output, while the other LSTM is used as decoder which take compressed vectors as input then gives

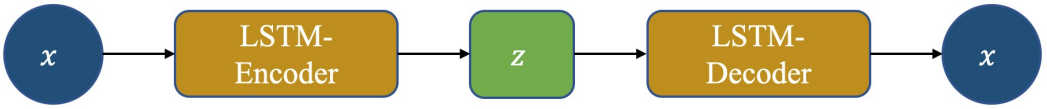


FIGURE 2.4: architecture of variational generative model.

reconstructed trajectory as output. The model structure is shown as figure 2.4, which follows a Variational Autoencoder (VAE) framework. The training process of this model follows a semi-supervised learning scheme, which means that the input and the output is the same. The loss function of this model is:

$$l_i(\theta, \phi) = -E_{z\ q_\theta(Z|X_i)}[\log p_\phi(x_i|z)] + w * D_{KL}(q_\theta(z|x_i)||p(z)) \tag{2.11}$$

Where θ, ϕ are parameters of decoder and encoder respectively. The first term of the loss function is the reconstruction accuracy measurement, Euclidean distance is often used for measure the reconstruction error. The second term is the KL-divergence of the approximate posterior distribution with a unit Gaussian Mixture. We assume the posterior distribution is a Gaussian distribution because we want to find a simple yet efficient latent space for real urban trajectories. As discussed in the above, Variational Autoencoder can build a hidden space which follow Gaussian distribution to approximate the real distribution of the observed trajectories. The reason for a constructing a hidden space which follows a Gaussian distribution is that by learning the parameters of the Gaussian distribution representing the input observed trajectories, we can sample from the distribution and generate new samples of trajectory. The ability for constructing hidden space following a Gaussian distribution is exactly what we want in the variational generative model. However, the Variational Autoencoder lack the ability of tackling sequential data, which is the main limitation.

The seq2seq model framework usually use several Recurrent Neural Networks as encoder and decoder. Therefore, a seq2seq model can handle sequential data without difficulties, but the hidden space C is not well constructed. We can regard the seq2seq model as a sequential Autoencoder. By doing that, it is natural to consider that if we combine Variational Autoencoder and seq2seq model as figure 2.4 shows, we can combine their advantages. That means the variational generative model is well-designed generative model for sequential data.

Let $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_t})$ denote a high dimensional sequence, such as a trajectory of human mobility wit t steps. We use a LSTM neural network as recurrent encoder to capture the information of the input trajectory $\mathbf{x}$. Then we will obtain a series of hidden state s_t, and a series of output o_t. In actual case, what we really care about is the final output o rather than a sequence of output value o_t. Since we only keep the final output, we can obtain a intermediate non-sequential vector o to represent the information captured from the input sequence using this recurrent encoder. After intermediate vector o is obtained, we treat this vector as the input of the Variational Autoencoder part. Then we can write the joint probability of the model as $p(o, z) = p(o|z)p(z)$. $p(z)$ is a prior latent distribution, and $p(o|z)$ is the likelihood. Then we need to calculate the posterior latent distribution $p(z|o)$ given observed data:

$$p(z|o) = \frac{p(o|z)p(z)}{p(o)}$$

by marginalizing out the latent distribution:

$$p(z|o) = \frac{p(o|z)p(z)}{\int p(o|z)p(z)dz}$$

This is a exponential time-consuming process. Therefore, variational inference approximates the real posterior distribution with a family of distribution $q_\lambda(z|o)$. Usually, we choose q to follow a Gaussian distribution, then λ would be the mean and variance of the latent distribution $\lambda = (\mu, \sigma)$. Kullback-Leibler divergence is used for measuring the information lost when using q to approximate p, the optimal approximate posterior is thus:

$$q^*_\lambda(z|o) = argmin_\lambda KL(q_\lambda(z|o)||p(z|o))$$

In the VAE model, we parametrize approximate posterior $q_\theta(z|o)$ using an inference network, approximate likelihood $p_\phi(o|z)$ using a generative network. Then the loss of the model will be:

$$loss = -E_{q_\theta(z|o)}[logp_\phi(o|z)] + KL(q_\theta(z|o)||p(z))$$

Finally, we use another LSTM neural network as recurrent decoder to reconstruct the trajectories of human mobility, $\mathbf{x}$ from parameters in learned latent distribution.

TABLE 2.1: summary of daily statistics

Total records	6,137,308,784
Total daily user IDs	1,168,592
Total route IDs	2,507,308
Average records	197,977,703/day
Average daily user IDs	38,632/day
Average route IDs	81,791/day

2.3 Experiments

2.3.1 Description of raw data

The data we used for this research is navigation locational data, which is collected when vehicles were using navigation application. The coordinate system of this GPS data is WGS84, and the records of the locations cover all over Japan. However, owing to some reasons, such as privacy protection, we can only use one month records which is from Oct 1, 2015 to Oct 31, 2015. Besides, the ID of the users were deleted, so the privacy is protected well. We can only get the information of the ID of each navigation route to distinguish different trajectories. Our data contains the information of:

1) Daily user ID: a random unique ID of a vehicle in a day;

2) Route ID: the unique ID of each navigation trip;

3) Timestep: the recorded time of current location;

4) Longitude and Latitude: the value of longitude and latitude after conducted map matching;

5) Sensor longitude and Sensor latitude: the value of raw records of longitude and latitude.

To get an intuitive image of the data we used, a visualization of the GPS data in selected Tokyo area is given as follows:

The figure 2.5 given is drawn using the recorded locational points. Since the records is dense and map matched, the points can shape lines and infer the road map perfectly. Moreover, we can imagine intuitively, more vehicles drive in major road than those drive in a small road, thus we can see that the lines of major roads are thicker than small roads. In summary of the human mobility trajectory in raw data set, we get a simple table 2.1:

FIGURE 2.5: Distribution of navigation GPS points of NAVITIME

2.3.2 Data preprocessing

The navigation GPS data we used in this research is a really big data and containsa wealth of sequential information. However, it is very difficult to handle such big data, we must do some data preprocessing for this raw data then get a data set we want to utilize in our experiment. The aforementioned basic statistics of the navigation GPS data is all done by coding using python. Since the whole data is as huge as 1.2 terabyte, divided into 938 common-separated values (csv) files, conducting statistics on such big data is very hard time-consuming work. To improve the efficiency of basic statistics, we use parallel computing to make full use of central processing units of my machine. Thus, the computing time is reduced to one sixth and save lots of time. We use the "haversine" formula to calculate the great-circle distance between two points, which is the shortest distance over the earth's surface.

$$a = \sin^2(\delta\phi/2) + \cos\phi_1 \cos\phi_2 \sin^2(\delta\gamma/2)$$

$$c = 2a\tan(\sqrt{a}\sqrt{(1-a)})$$

$$d = Rc$$

Where ϕ is latitude, γ is longitude, R is earth's radius (mean radius is 6,371 km). Then, we get distance delta between each two points using above algorithm. By summarizing the distance delta of the same navigation trip, we can finally get the traveling distance of all trajectories in the navigation GPS data. Another processing is that we also compute the time interval between each two points, although it is not used in aforementioned basic statistics, but it will be useful for the experiment. As the time interval of the raw data is not fixed, which means that it will lead to some potential difficulties to further use. To simplify the data structure of the data which we will use in the experiment, we conduct a linear interpolation to the navigation GPS data to make the time interval of the records fixed. The reason for a linear interpolation is two-fold: 1) simplify the data structure; 2) obtain trajectories in a specific length. The navigation GPS data is not intuitive for those who are not familiar with trajectory data, so the visualization of the navigation GPS data is necessary. The visualization tool is called mobmap developed by Satoshi Ueyama, a researcher from our laboratory. In this paper, we use mobmap to visualize both the raw GPS data and the output results of our proposed model to make the data and result more intuitive.

For creating the data set used in our experiment, not all raw data is necessary as the size of the raw navigation GPS data is too big. Instead, we choose a selected Tokyo area, longitude from 135.5 to 139.9 and latitude from 35.5 to 35.8. Also, it is not necessary to use the whole month GPS data since the most navigation distance is shorter than 50,000 meters and will be ended in one single day. Since most of navigation trip will last hours, it is natural to get one hour's data to conduct the experiment. We select the records of from 10 am to 11 am October 1, 2015. The data set contains more than 2,000 trajectories, which has fixed time interval.

2.3.3 Experimental settings

We make a brief description about the general process of how to train the VAE model. The first step is preparing training data. The input data we used in the experiment is navigation GPS data which contains trip ID, longitude, latitude, and timestamp. However, the raw data should be preprocessed before the training process. The data preprocessing of linear interpolation is done to simplify the input data, by forcing the trajectories have fixed timestamp. Therefore, the input only contains information about

longitude and latitude but can still represent the dynamics of the trajectories. We then use several LSTMs as a recurrent encoder which aims to capture the salient features of the input sequential data. LSTMs return an output in every step, which means that the output could also be a sequential output. However, in the VAE model, a non-sequential output, which we make it a intermediate vector, is better. This intermediate vector captures the salient features of the input trajectories while keeps a non-sequential data structure. We want the intermediate vector to be non-sequential since the custom variational autoencoder has no ability to handle the sequential data. After the intermediate vector is given by the recurrent encoder, it will be the input of the custom variational autoencoder. This layer aims to build the latent space which can capture the features of the input and follow a Gaussian distribution at the same time. The output of this layer is mean and logarithm variance which are used for constructing the latent space which follows the Gaussian distribution. The final output of this layer is sampled from this latent space, and it will be the input of next recurrent decoder. The latent vector should be repeated several times to match the length of the output trajectories. Then the recurrent decoder consisted of several LSTMs will reconstruct the output trajectories using aforementioned latent vector. Reconstructed trajectories should be as similar as possible comparing with input original trajectories by minimizing the loss function. At the same time, the latent Gaussian distribution should also be as simple as possible to make the VAE model robust.

We use aforementioned data to conduct experiment. Mean Distance Error (MDE) between real trajectories and generated trajectories is used for evaluating the performance of the VAE model with different parameter settings:
(1) Short sequence and long sequence, of which length is 6 and 20 respectively, as input of the VAE model to test the ability for tackling long sequence of the model;
(2) The dimensionality of hidden space is set to be 8, 12, 16 respectively to test the performance of the model for different dimensionality of hidden space;
(3) Three kinds of input (values of coordinate only, grid ID only and combination input of values of coordinate and grid ID) are tested.
The results are given in next section.

TABLE 2.2: VAE Loss

Loss	both input	coordinate input	grid ID input
6 steps, 8 latent dimension	0.017318176	0.018159691	0.017484304
20 steps, 8 latent dimension	0.027957876	0.02932067	0.027624224
6 steps, 12 latent dimension	0.017548803	0.018433879	0.01732461
20 steps, 12 latent dimension	0.027994325	0.030799899	0.029310457
6 steps, 16 latent dimension	0.017232143	0.018182858	0.017890416
20 steps, 16 latent dimension	0.029631174	0.031207314	0.03502047

2.3.4 Results and Visualization

We use two datasets as our training set of VAE model. One dataset is 2,000 trajectories of which length is all set to be 6, and the other one is 2,000 trajectories of which length is all set to be 20. The two data set is all chosen from the same raw dataset, but with different length of every sequence. Respectively, we train the VAE model using these two datasets, changing the parameters which controls the dimensionality of the constructed latent space, and inputs.

The values of loss in different VAE models have been summarized in the table 2.2. The values of the loss is calculated using aforementioned formula:

$$loss = -E_{q_\theta(z|o)}[logp_\phi(o|z)] + KL(q_\theta(z|o)||p(z))$$

The values in the table is given by the loss of final step's training. The smaller the value is, the better the results of the trained VAE should give theoretically.

Owing to the lackness of exited generative model for trajectories of human mobility, we evaluate our results just using the designed Mean Distance Error (MDE).

$$E_j = \frac{\sum_{i=1}^{N} dis(l_ij, \hat{l}_ij)}{N}$$

where $dis(a, b)$ calculate the distance of point a and point b using their coordinate values; l_ij is the groundtruth, and $\hat{l}_ij$ is the outputs of the VAE model.

In figure 2.6, we also give a visualization of four true trajectories chosen from groundtruth and its corresponding reconstrcuted trajectory. From the figure, we can see that the driver moves from south-east to north-west in about 20 minutes. Therefore, the locations of true record and reconstrcuted record in every 5 minutes is given to show accuracy

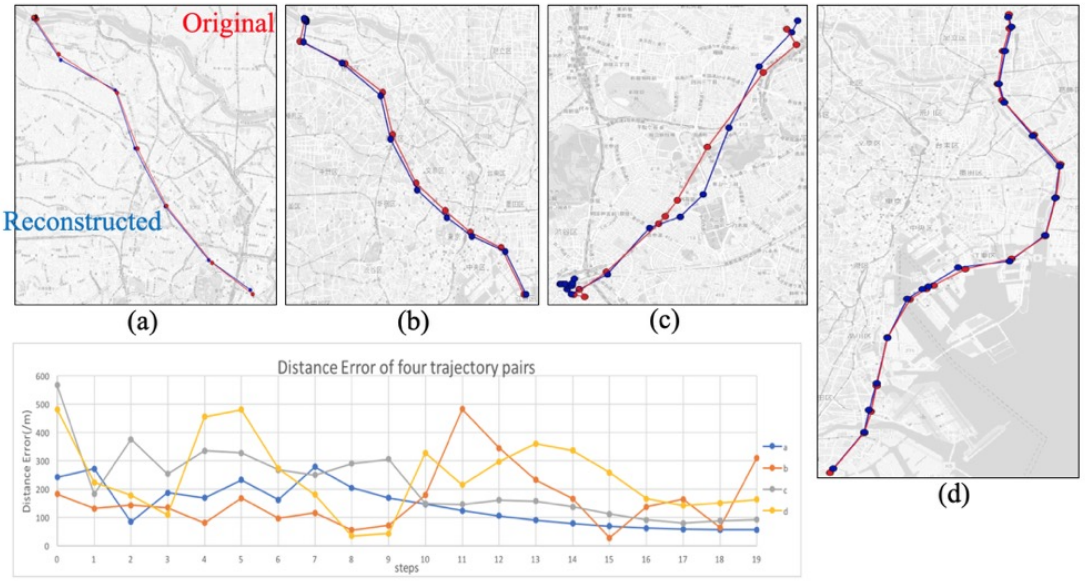


FIGURE 2.6: Reconstruction error of generated points (/Meter). The reconstruction accuracy is acceptable.

of the results in a intuitive way. Instead of just giving a visualization of the single trajectory, a quantitative measurement is given by calcaluting the distance of two points in every step. The units of the distance error is meter. We will use this result for explaining the limitation of the VAE model.

2.4 Conclusion

2.4.1 Discussion

Besides, it should be mentioned that training the VAE model using long sequences as inputs is more time-consuming. Therefore, The epochs of iterations should be carefully considered to reduce the computation. We set epochs of iterations to be 1,000, which make sure that the model is trained fully. By adding regularizers in our neural network layers, we can avoid overfitting. Table 2.2 indicates three points: (1) in general, the loss values of training using both coordinate values and grid ID as input is smallest, follows training using grid ID as input and using coordinate values as input; (2) the loss values of higher dimensional latent space is often larger than those of lower dimensional latent space; (3) the loss values of long sequences is larger than those of short sequences. A reasonable explanation of aforementioned phenomenon is that the loss function of

this VAE model is designed as the combination of reconstruction errors and Kullback-Leibler divergence of approximated posterior distribution and unit Gaussian distribution. Therefore, the phenomenon of that the loss value of training long sequences with a higher dimensional latent space is larger than others can be easily explained. Since long sequences have 20 steps, it is likely that the sum of 20 small loss is greater than the sum of 6 small loss of short sequences, which have 6 steps. That will cause the greater reconstruction error of loss function when training long sequences. The situation for higher dimensional latent space is almost the same. Higher dimensional latent space is likely to have greater values of Kullback-Leibler divergence, which is also a part of loss when training the VAE. Since that the approximated posterior distribution constructed in latent space of VAE model is aimed to capture as many features of input training data as possible while in a limited capacity. When the input data is very complex, then the capacity of the latent space should be larger to be able to learn the features. If the dimensionality of the latent space is limited to be small, then it will lead to the lackness of ability of learning most of the features of training data. However, when we increase the dimensionality of latent space, the ability for learning features of the VAE model is increased indeed. But it could not be always the right strategy to increase the dimensionality of latent space since the information contained in training data is finite, which means a proper VAE model can learn most of the salient information contained in training data using a finite dimensional latent space. Therefore, when the dimensionality of latent space is too high, the story becomes to be that part of the latent space capture the most salient features, and rest of the latent space is used to deal with the redundant trivial information. In that case, a higher dimensional latent space achieve a performance just like a lower dimensional latent space or even worse.

Comparison pf foure individual trajectory and virtual generated trajectory is given by figure 2.6. Also, we calculate the distance error in every 3 minutes, which is shown as the bottom of figure 2.6. The tendency of the distance error is reducing with time, which means that reconstructing a moving trajectory is harder than reconstructing a staying object. In general, the results of evaluation using MDE shows that the reconstruction error of VAE model is smaller than 800 meters. Considering that the selected experiment area is about $33,000m \times 36,000m$, we think that the accuracy of results of the VAE model is enough to tackle the city scale problems. Actually, there is a trade-off between the accuracy of the reconstruction trajectories and the robustness of the ability to generate

resampled trajectories. As mentioned before, the loss function of the VAE model is consist of reconstruction error and Kullback-Leibler divergence. In practical training process, minimizing the reconstruction error will increase the accuracy of reconstructing input trajectories, while minimizing the Kullback-Leibler divergence will reduce the complexity of learned latent space. The goal of training the VAE model is to minimizing both reconstruction error and Kullback-Leibler divergence. However, there can be a trade-off between them as we usually add weight, smaller than 1, to one of them. When we want our model achieve higher accuracy in reconstructing trajectories, we add a small weight to Kullback-Leibler divergence to reduce the contribution of Kullback-Leibler divergence for the whole loss. Therefore, the training process become that we care less about the complexity of the learned latent space, just make sure the output reconstructed trajectories are as accurate as possible. In that case, we can get a model of which has a very good performance of reconstructing input trajectories but a very poor performance of generate resampled trajectories. In the other hand, if we add a big weight to Kullback-Leibler divergence, we aim to train a robust model of which latent space is as simple as possible. Therefore, we are likely to get a robust model which has a poor performance of reconstructing input trajectories. Both of the aforementioned model is not the ideal model we want. In overall, as discussed above based on the evaluation and visualization of the results, we think our model is trained in a balance way.

2.4.2 Limitations

We also want to make a brief discussion about the limitation of the current VAE model when handling the trajectories of human mobility. As shown in figure 2.6, a real trajectory and its corresponding reconstructed trajectory is given. The real trajectory is tortuous while the reconstrcuted trajectory is smooth. Although the reconstruction error is small, the output reconstrcuted trajectories of the VAE model seems to be a smooth approximation of tortuous real trajectories. A main limitation is that many points of reconstrcuted trajectories don't located in road network. implementing map matching to the generated trajectories may solve the problem, but we believe a better choice is that change the current coordinate and grid based model to a node based model. Another idea for this problem is changing the current resampling from Gaussian distribution strategy to resampling from historical trajectories.

Chapter 3

Retrieval-based Human Trajectory generation

3.1 Introduction

3.1.1 Background

We analyzed the advantages and disadvantages of generating human mobility data through a deep generative model directly in the previous chapter. The main limitation of directly using the deep generative model for trajectory generation is that the generated virtual human mobility trajectory data does not have geographic information restrictions. Therefore, although the newly generated virtual human mobility trajectory does not infringe on user privacy, it cannot be proven to be realistic in the real world which the trajectory falls perfectly on the road grid. However, we have also tried some map matching methods to generate virtual human mobility trajectories with no geographic information constraints to the road network. As discussed in the previous chapter, directly using map matching as a post-processing method will change the pattern similarity between ground truth human mobility trajectories and newly generated virtual trajectories. Based on the shortcomings of these previous practices, we consider the direction of improvement from the perspective of pursuing the authenticity of the trajectory.

Another issue worth considering is how we measure the authenticity of the virtual human mobility trajectory directly generated by the deep learning model. We realize that the authenticity of the human mobility trajectory is a more difficult indicator to quantify and measure than the similarity of the human mobility trajectory. Here we are taking the similarity of human mobility trajectories as an example. Many previous studies have put forward some quantitative measures. Their primary purpose is to establish whether different human mobility trajectories are similar to tackle the trajectory clustering problem. However, the authenticity of the virtual human mobility trajectory cannot be measured entirely by the trajectory similarity. It is because that even if we assume that the virtual human mobility trajectory data generated by the deep learning model, which is similar to the ground truth human mobility trajectory, has a high authenticity. We cannot deny that generated virtual human mobility trajectory, which is not similar to the observed human mobility dataset, is not real. We must remember that the human mobility trajectory data we can access is usually a low sampling rate with sampling bias data. Therefore, we do not have enough evidence to deny that the human mobility trajectory that is not similar to the observations we get does not exist in this world.

Suppose we insist on taking the traditional measure of trajectory similarity to measure the authenticity of the newly generated virtual human mobility trajectory. In that case, an essential prerequisite is that we need to assume that the actual observed human mobility trajectory data we can currently obtain is sufficient and can reflect the flow of people in the real world. This assumption itself is untrue, and it is in contrast to the purpose of our work because under this assumption. In this case, we do not need to explore a suitable method of estimating the human mobility trajectory data to solve human mobility trajectory data which does not fully reflect the real-world human mobility pattern. Nevertheless, in any case, we hope that the new virtual trajectory generated by the deep learning model is realistic enough, so we should use another idea to solve this problem.

In order to avoid the authenticity problem of the generated virtual human mobility trajectory data and at the same time refer to enough research on the similarity of human mobility trajectory methods, we propose a retrieval-based human mobility trajectory generation method. Our method is still based on the deep learning method and uses the deep learning model to capture essential features in the human flow trajectory.

3.1.2 Problem definition

We refer to the traditional definition and method of similarity of human mobility trajectory before and convert the problem of directly generating virtual human flow trajectory into a problem of using a deep learning model to construct trajectory similarity. We define it as historical human mobility trajectory data for the entire human mobility trajectory data we can currently get. This historical human mobility trajectory data acts as a database. In this database, we only keep the trajectory data and delete all other information that may infringe on personal privacy. One of our assumptions here is that although we only have a small part of the real human flow trajectory data, we can still get a trajectory database with sufficient information if we continue to sample for a long time. Assuming that we already have a small part of the human mobility trajectory data observations, we need to generate more virtual human flow trajectories. We can use some of the trajectories in this historical trajectory database as supplementary trajectories. Therefore, we turn the problem into selecting a suitable supplementary trajectory from the historical trajectory database to make the small part of the observed human flow trajectory data more in line with the current real human flow pattern.

3.1.3 Research objective

In this chapter, our research objectives are as follows:
1) Conduct experiments to evaluate the change of citywide human mobility pattern when using map matching technique as postpocessing;
2) Propose a search-based method to generate human flow trajectory data to avoid the problem of imperfect authenticity encountered when directly generating virtual human flow trajectories;
3) We compare the human flow trajectory similarity method established by the proposed trajectory feature extraction method based on the Encoder-Decoder deep learning framework with the traditional method;
4) Carrying out numerical experiments to verify that our proposed method has obtained better results

3.2 Map-matching as post-processing

This section aims to evaluate the human mobility pattern change between generated virtual trajectories and virtual trajectories after map-matching.

3.2.1 Framework

For simplification, we use a statistical method of probability of migration for virtual trajectory generation, which can be written as the following equation: [50]

$$< T_{ij} >= T_i \frac{m_i n_j}{(m_i + s_{ij})(m_i + n_j + s_{ij})} \tag{3.1}$$

where T_i is the total number of commuters that start their trip from location i, which can be calculated by:

$$T_i = m_i(\frac{N_c}{N}) \tag{3.2}$$

where N_c is the totcal number of commuters and N is the total population in the city. Besides, m_i, n_j, s_{ij} indicates the population of grid i, j, from grid i to j respectively. Therefore, the first step is to convert the real human mobility trajectories to grid-based sequences. By implementing the above equation, we got generated coarse-grained virtual trajectories. Then, we know that the generated coarse-grained trajectories should be processed using the Shortest path algorithm with map matching technique to produce virtual finer-grained trajectories in the road network. The entire process of producing those virtual trajectories is shown in the figure. Our objective in this section is that we want to evaluate human mobility pattern changes after this process.

3.2.2 Experiments

Data description We used a month's taxi navigation GPS track, and the data range is a rectangular area within a city that is roughly 8kmx8km. The data collection interval we used is 2 4s, which shows that the data was initially high-quality trajectory data. At the same time, the original data points are all on the road network.

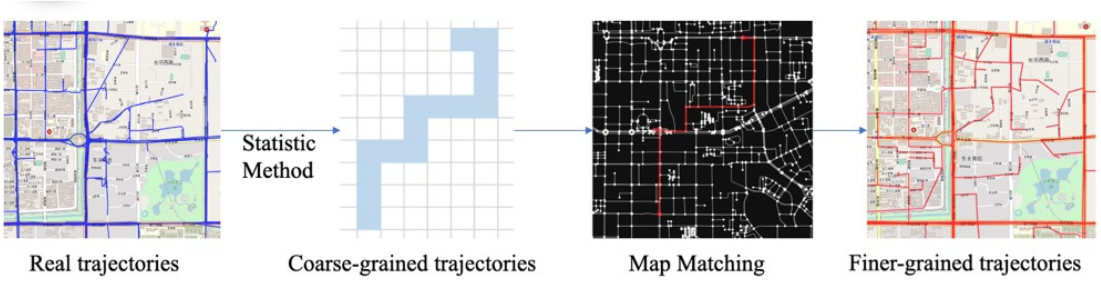


FIGURE 3.1: Framework of using shortest path algorithm with map matching technique as postpocessing of virtual human mobility trajectories.

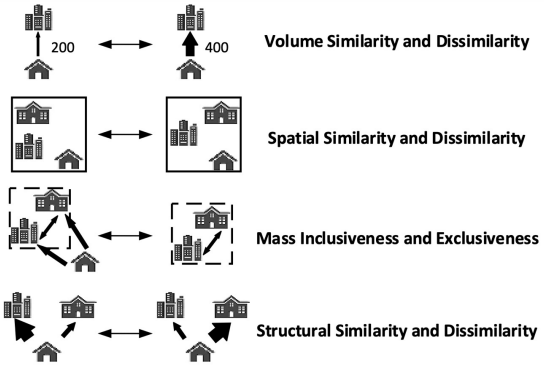


FIGURE 3.2: The above four figures give an illustration of four metrics of human mobility trajectory similarity.

Metrics for assessment Mean Square Error (MSE) and Mean Absolute Error (MAE) is used in this section, which can be written as:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \quad MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i| \tag{3.3}$$

We also use some trajectory similarity metrics proposed recently shown in the figure 3.2 [51].

3.2.3 Results

At first, we will show a visualization for comparison between ground truth trajectories used in this experiment and finer-grained virtual trajectories after processing of shortest path algorithm with map matching technique. We notice some differences between the two figures: 1) the trajectories distribution is different, meaning there are some virtual trajectories in some places with no ground truth trajectories. 2) shortest path

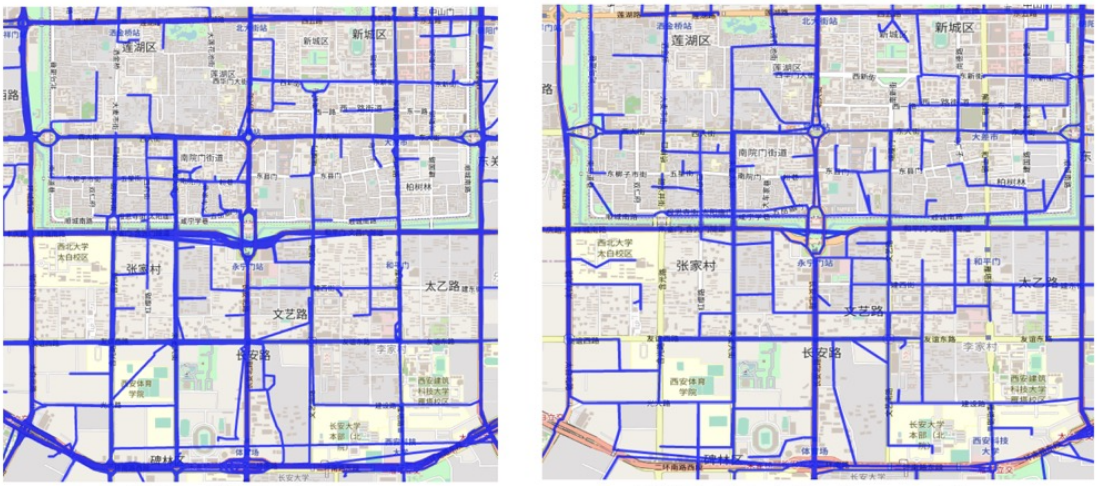


FIGURE 3.3: Visualization of ground truth trajectories and virtual trajectories after map matching.

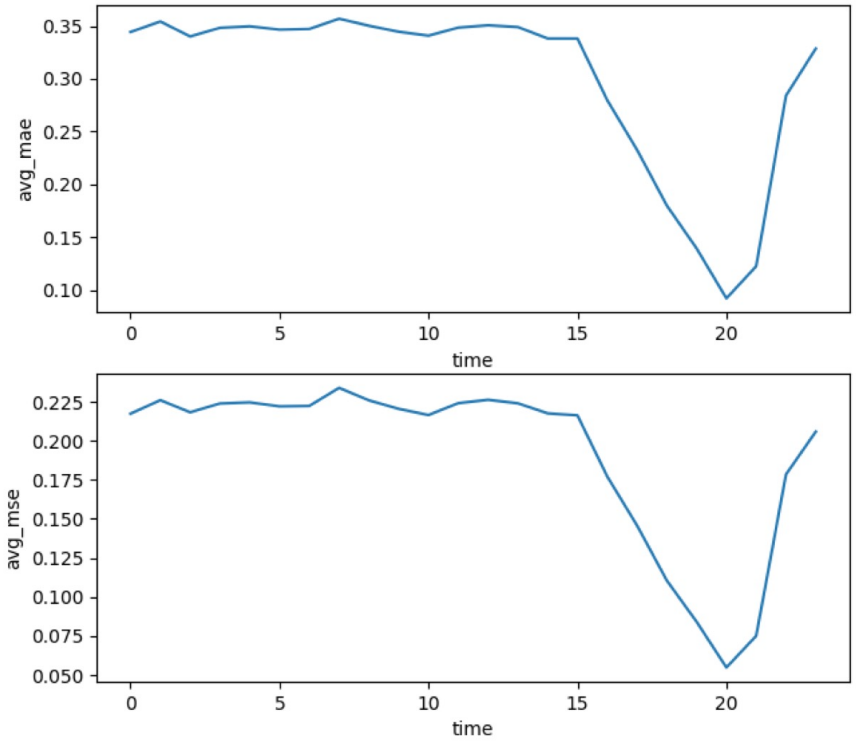


FIGURE 3.4: Average MAE (up) and MSE (bottom) at different time.

algorithm with map matching technique introduces some error in virtual finer-grained human mobility trajectories.

Since the predicted trajectory and the real trajectory are equivalent to one hour in length, and there are only the starting point and the endpoint, and the trajectory information in the middle part is ignored, the trajectory is split for more detailed trajectory evaluation.

Split the trajectory for 20min and 30min respectively (equivalent to sampling at 1/3, 1/2 of the trajectory grid list) to split the original trajectory into 2 or 3 sub-trajectories, and then use these sub-traces trajectories to evaluate the quality of the production. (The true value and predicted value are split in the same way

Table 3.1: Performance of Shortest Path with map matching.

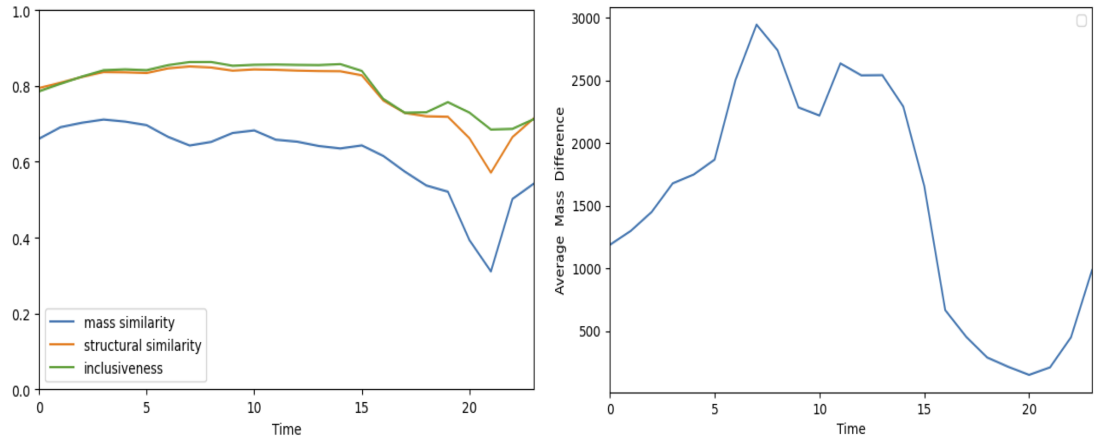
	mass difference	mass similarity	inclusiveness	structural similarity
60 min	1542.21237	0.61311628	0.80383485	0.78724814
30 min	4996.70296	0.57481026	0.74331417	0.76080039
20 min	7738.34139	0.54148302	0.72683068	0.73145890

In short, the conclusion is that the more refined the trajectory is split, the worse the effect displayed by the evaluation index.

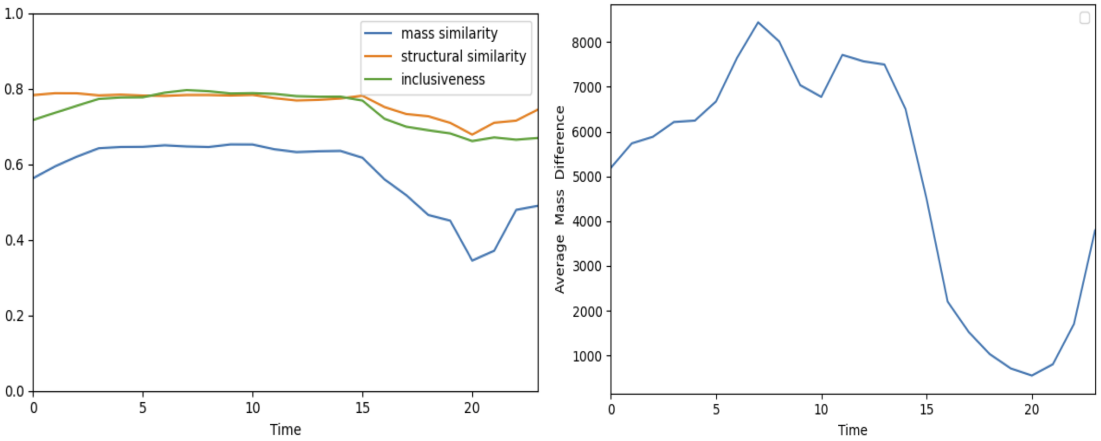
3.2.4 Discussion

The data is a kind of navigation data of trips. People drive mainly to the core areas with large numbers of people or core residential areas in cities. At the same time, the time when the number of cars is the largest is about 7:00 16:00. According to the human migration formula, people are more inclined to go to the grid with more people. The probability is significantly reduced when The distance of the grid with many people is significant. So the reflected movement trend must be related to the number of cars in the grid (that is, the predicted trend must be $A \to B$ if the movement of people who go to work in the morning is $A \to B$ when people drive a car to return home in the evening, $B \to A$ is required. The predicted value is still $A \to B$). This causes the number of cars to decrease, and the deviation of the predicted trajectory in the afternoon and evening of the return journey gradually increases.

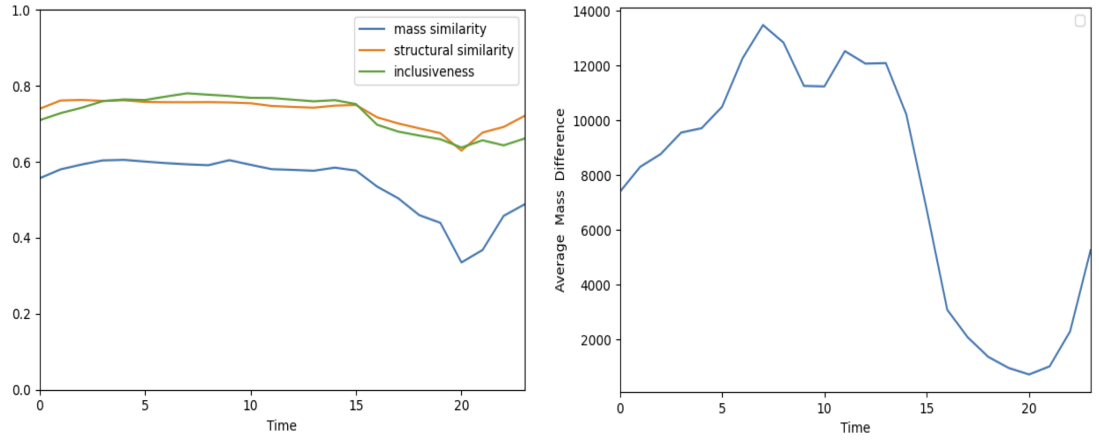
Since the current map matching is based on the shortest path, the shortest path may not be reasonable in the actual situation. For cars, the data acquisition system will also consider it when planning the path. Actual traffic conditions, road conditions, and other information, it is entirely reasonable for a car to move a long distance to pursue the shortest time. So if the prediction is correct for the starting point and the endpoint, the result is correct without splitting the trajectory at this time. If the trajectory is split, it is likely to cause an error, which will result in a drop in the evaluation index. At the same time, the greater the number of trajectories split, the greater the number



(a) 60 min



(b) 30min



(c) 20min

FIGURE 3.5: Performance evaluated by four human mobility trajectory similarity metrics.

of erroneous trajectories, which leads to a further decline in the evaluation index. That is, if the transfer vector used for evaluation is correct when it is not split, it is possible that if it is split into two, then two transfer vectors will be wrong, and if split into three, three transfer vectors will be wrong, resulting in a decline in the evaluation index.

3.3 Retrieval-based model

3.3.1 Preliminary

Bidirectional Long-Short Term Memory Deep neural networks which are composited mainly feedforward fully connected neural networks are powerful but not appropriate for sequence data such as time-series data or natural language. They are outstanding to map input data to discrete output or continuous variables but not sequence to sequence mapping. Sequence-to-sequence (seq2seq) model uses two Long Short-Term Memory (LSTM) models. One LSTM learns vector representation from the input sequence of fixed dimensionality, and another LSTM learns to decode from this input vector to the target sequence. LSTM is a variant of recurrent neural networks that solves long sequences using different gates. Seq2seq model was recently proposed and demonstrated the excellent result of Natural Language Processing (NLP) [52–54]. This model proved to be more effective than previous methods at NMT and is now used by Google Translate.

LSTM was designed to combat vanishing gradients through a gating mechanism by Hochreiter and Schmidhuber in 1997 [49]. The ability to model long-term dependencies is improved in LSTM thanks to the gating mechanism. However, a single LSTM is insufficient to learn the context information because the hidden space is learned heavily depends on the input information before while some future information is ignored. Schuster and Paliwal proposed a bidirectional recurrent neural network structure to lean the context information from both forward and backward directions in 1997 [55]. We use bidirectional LSTM to achieve spatiotemporal feature extraction of human mobility

trajectory. Bidirectional LSTM can be mathematically described as:

$$\begin{aligned}
\overrightarrow{h}_t &= H(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \\
\overleftarrow{h}_t &= H(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}) \\
h_t &= [\overleftarrow{h}_t \overrightarrow{h}_t]
\end{aligned} \tag{3.4}$$

We extract the intermediate vector with the fixed dimension as the input of Variational Autoencoder to approximate the latent distribution of spatiotemporal features. Still, the raw GPS trajectories with different lengths are difficult for batch processing, and the LSTM model will go down with the length of the input sequence. Bearing these two concerns in mind, we slice the historical trajectories into a fixed length , the process of which can be written as:

$$\hat{tr} = tr[i : i + L]|i = 0, L, ..., tr.length - L \tag{3.5}$$

Where $tr[i : i+L]$ is the subsequence of tr with the indices j ranging from $i \leq j \leq i+L$.

K-dimensional tree The k-dimensional tree, which is a multidimensional binary search tree, was invented by Jon Louis Bentley in 1975 [56]. Here we give a brief introduction of the k-d tree data structure. Let $P = \{p_1, ..., p_n\} \subset \mathbb{R}^d$ be a finite data set with $p_i = (p_i^1, ..., p_i^d)^T \in \mathbb{R}^d$ for $i = 1, ..., n$. A k-d tree for P is defined recursively. An empty tree or a tree with only one node which contains the only one data point will be built if P is empty or contains only one data point respectively. Otherwise, it is determined in which dimension $d' \in [d]$ the data set has the largest spread.

That is, d' is chosen such that there are two points p_i, p_j with $|p_i^{d'} - p_j^{d'}| \geq |p_l^{d''} - p_m^{d''}|$ for all $d'' \in [d], l, m \in [n]$. For any number $n \in N$ of ordered points, we define the median to be the point with index $[\frac{n}{2}]$. Now, we should find the median of the points p_i according to a sorting along this dimension, i.e. $p_{i_q}^{d'} \leq ... \leq p_{i_{\frac{n}{2}}}^{d'} \leq ... \leq p_{i_n}^{d'}$, denoted by $q = p_{i_{\frac{n}{2}}}$. After finding the median, a hyperplane $H = \{x \in \mathbb{R}^d | x^{d'} = p_{i_{\frac{n}{2}}}^{d'}\}$ is introduced, which splits the set P into two subsets:

$$P_1 = \{p_{i_1}, ..., p_{i_{\frac{n}{2}-1}}\}$$

$$P_2 = \{p_{i_{\frac{n}{2}+1}}, ..., p_{i_n}\}$$

with P_1 containing at most one point more than P_2. A node is created, holding q and H. The node is given the results of recursively processing P_1 and P_2 as children and then it is returned.

3.3.2 Framework

The basic idea of building a retrieval-based trajectory generator is that we use VAE as a feature extractor because we have shown its capability of learning the latent vector space given human trajectories as inputs. Since directly generating new trajectories by sampling from latent vector space causes some limitations mentioned above, we consider the posterior distribution parameterized by vectors corresponding to human trajectories as index. Then, this index is used for retrieving more trajectories from historical dataset of which the distance of index is under a threshold with query trajectories.

A query trajectory dataset is arbitrary selected, which consist in all human trajectories at current time we may be interested in. From information retrieval perspective, a historical trajectory dataset can be regarded as a knowledge base for providing some supportive information for a query trajectory. We define a query trajectory dataset as $traj^q$, and a historical trajectory dataset as $traj^h$. It should be mentioned that some query trajectories in $traj^q$ and some historical trajectories in $traj^h$ can be sliced from some long original trajectories, so they have more correspondence even they are different in some GPS points.

Given a query trajectory dataset $traj^q$, which usually consist in only a small number of human trajectories, can hardly reflect the movement pattern especially in some rural area. For example, we may observe enough human trajectories in a high way to briefly estimate some statistics to describe the traffic situation of this high way. However, there are lots of small streets which we cannot ignore when a more fine-grained estimation is necessary. In such small streets, no observation can be found sometimes considering the low sampling rate of the dataset. For instance, at 8 clocks, we observe 5 cars go through the street, while at 9 clocks, we observe 0 car, assuming we know the low sampling rate of our trajectory dataset is 1%, we cannot conclude that there are 500 cars go through the street at 8 clocks, while no single car at 9 clocks. Therefore, we want a reasonable approach to recover a bigger dataset which has more reasonable trajectories given a small dataset.

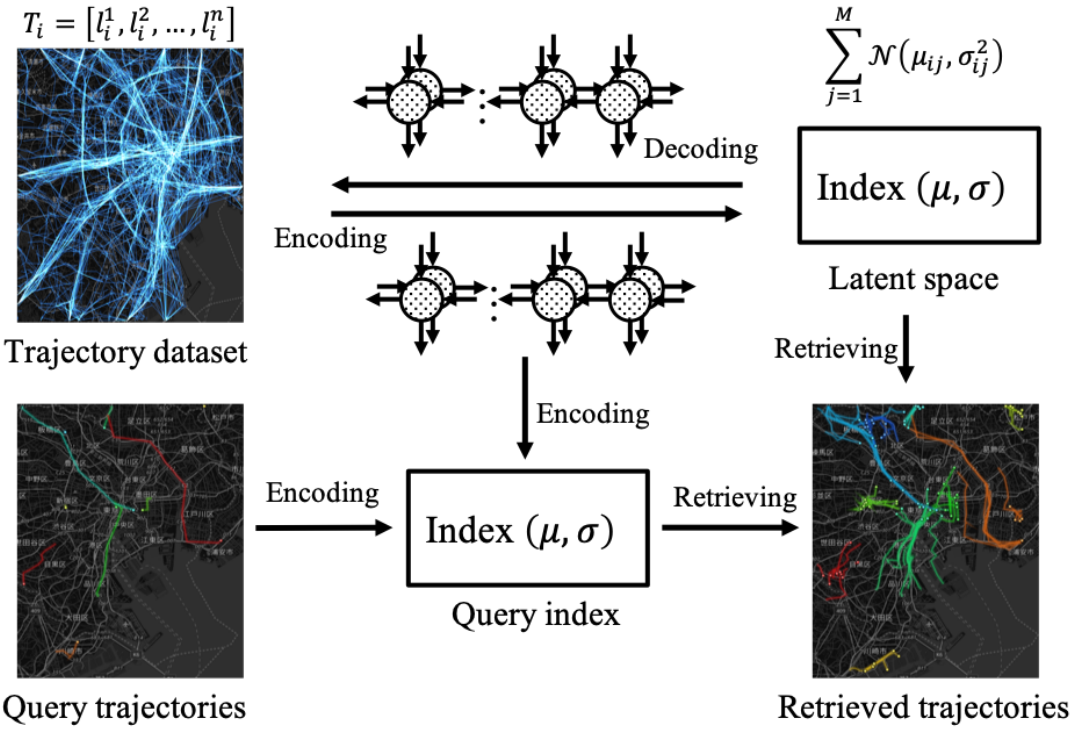


FIGURE 3.6: Framework of retrieval-based human mobility generation model.

We give a framework of our retrieval-based trajectory generator in 3.6. At first step, we train a VAE model using a large historical trajectory dataset. Then, we regard this large trajectory dataset as a database in which every trajectory indexed by their corresponding latent vector. It is also known that the latent vector of each trajectory forms a Gaussian mixture.

In next step, our goal is to recover the large trajectory dataset given a small limited query trajectory dataset. We use the encoder in of trained VAE to get its corresponding vector index of each query trajectory, then we use k-dimensional tree method to find some nearest points of query index in historical dataset. Finally, we select those urban human trajectories using these nearest points to recover a large trajectory dataset.

3.4 Experiments

3.4.1 Data description

This study utilizes the navigation GPS dataset, a dataset of navigation route GPS records of different types of cars with user ID intentionally deleted to preserve users' privacy. In this dataset, we can only recognize each navigation route's ID, so we cannot track the specific users in the long term, although one-month data was provided. For each trajectory of a single navigation route, the location of each coordinate point is map matched to the road network, and its timestamp is just about 2 or 3 seconds. It is a very high-quality navigation GPS trajectory data. However, bearing the limited capacity of LSTMs, we decided to crop the trajectory data into a fixed 1-hour length, with the timestamp as 1 minute, so the length of every single trajectory is 60 steps. To quantitatively experiment, we use the 1-hour length of trajectory dataset of which timestamp is 2015-10-31 11:00:00 as the query trajectory dataset and those trajectory datasets before as historical trajectory dataset.

Two straightforward representations exist for the trajectory cells, the one-hot representation and the coordinates of the GPS points. For the case of raw GPS records that contain random noise, an ingenious method of pre-training cell presentations was proposed to create the context for each cell instead of using the two representations as mentioned above. However, in our experiment, we assume the coordinates of every GPS point are exactly correct since they were all matched to the road network. Under this assumption, we use coordinates of GPS points for the presentations of trajectory cells to reduce the complexity of the experiment since the coordinates of GPS points naturally encode the spatial proximity for the cells of trajectory.

3.4.2 Baseline methods and Metrics

We compare our method with the previous t2vec method [57], which is a deep learning-based trajectory similarity method, and other four traditional trajectory similarity methods for measuring the trajectory similarity, namely Dynamic Time Warping (DTW), Edit Distance on Real sequences (EDR), Edit Distance with Projections (EDwP), and

Longest Common Subsequence (LCSS). Since we use map-matched GPS records of trajectories, we do not have to learn cell presentations through pre-training, so we use a simplified version of t2vec as a baseline method. Besides, we use LSTMs and bidirectional LSTMs to replace RNNs in t2vec as we need LSTMs' capacity to handle longer sequences. The method of t2vec is the first solution based on deep learning for creating representations of trajectories that are used as a trajectory similarity method. This method is at least one order of magnitude faster than other traditional trajectory similarity methods and has higher accuracy. While DTW, EDR, EDwP, and LCSS are some very classic trajectory similarity measurements. DTW is first introduced to measure time-series data, including this method as a baseline method. Also, both LCSS and EDR are two of the most widely used methods for analyzing spatiotemporal data. EDwP is the state-of-the-art method for measuring the similarity of non-uniform and low sampling rate trajectories. Moreover, we compare with the LSTMs to show bidirectional LSTMs' ability to learn the spatiotemporal features of trajectories.

We given three metrics for the assessment of different models for retrieving human mobility trajectory data for reconstruct the citywide human mobility pattern. CityEMD is defined as follows:

$$CityEMD(t) = \sum_{o} dist(p(d|o), \hat{p}(\hat{d}|o)) \tag{3.6}$$

Where $d, \hat{d}$ is ground truth, retrieved locations of subjects originated from o repectively.

Another metric called hard matrch ratio is defined as ratio of retrieved trajectories which share the same route ID with training data:

$$\sigma = P(\{B|B^{ID} \subset A^{ID}|A, H) \tag{3.7}$$

Because a complete trajectory of navigation has a unique route ID, so if a sub-trajectory has the same route ID with another one, they are preprocessed from the same long complete trajectory. So if we can retrieve more this kind of trajectories, we consider our model has the better ability to reconstruct the dataset.

The other one is recover ratio.

$$\xi = P(A|A - B, H + B) \tag{3.8}$$

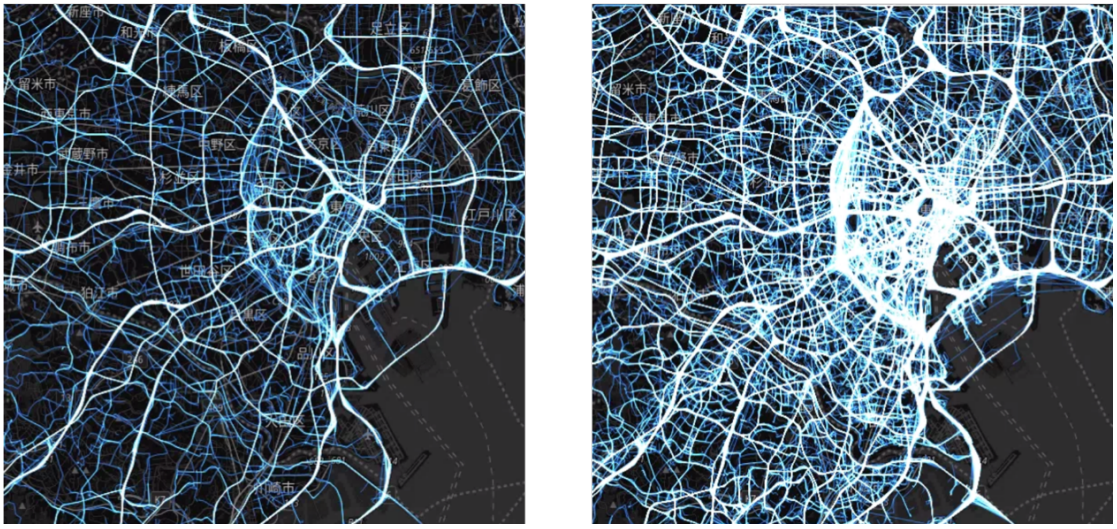


FIGURE 3.7: Visualization of ground truth human mobility (left) and retrieved human mobility (right).

We using a subset of target trajectory dataset which is $A - B$, and throw subset B into historical dataset H, if we can retrieve more trajectories belongs to B from historical dataset $H + B$, we consider our model has a better ability to recover the original dataset.

3.4.3 Results

Here we show some experimental result. Figure 4 shows that the query trajectories lack some details in rural areas in Tokyo, while recovered trajectory dataset has more details even in those area. Since all retrieved trajectories are selected in historical database, so it is a very reasonable that these trajectories could be shared by someone else even though they are not captured by collected dataset due the low sampling rate.

We also give some quantitative comparison with some baseline methods. The metrics of evaluation is CityEMD, hard match ratio, and recover ratio. CityEMD is a kind of distance proposed by previous researcher that measure the similarity of large area human mobility pattern. Hard match ratio is the ratio of retrieved trajectories which share the same route ID with part of known training data. Recover ratio is a probability of recovering the known training dataset.

Figure 3.8, 3.9, and table 3.2 show that our methods outperform other traditional trajectory similarity methods. It means that our retrieval-based trajectory generator has better ability of constructing a large trajectory dataset which is more similar with real world.

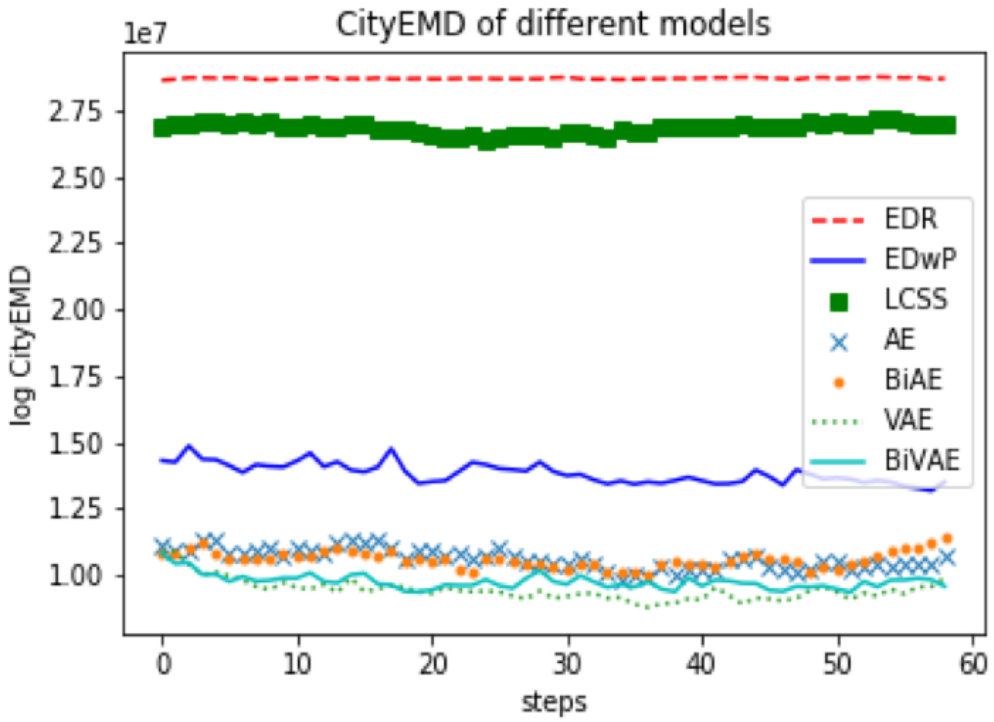


FIGURE 3.8: Comparison between different methods using CityEMD.

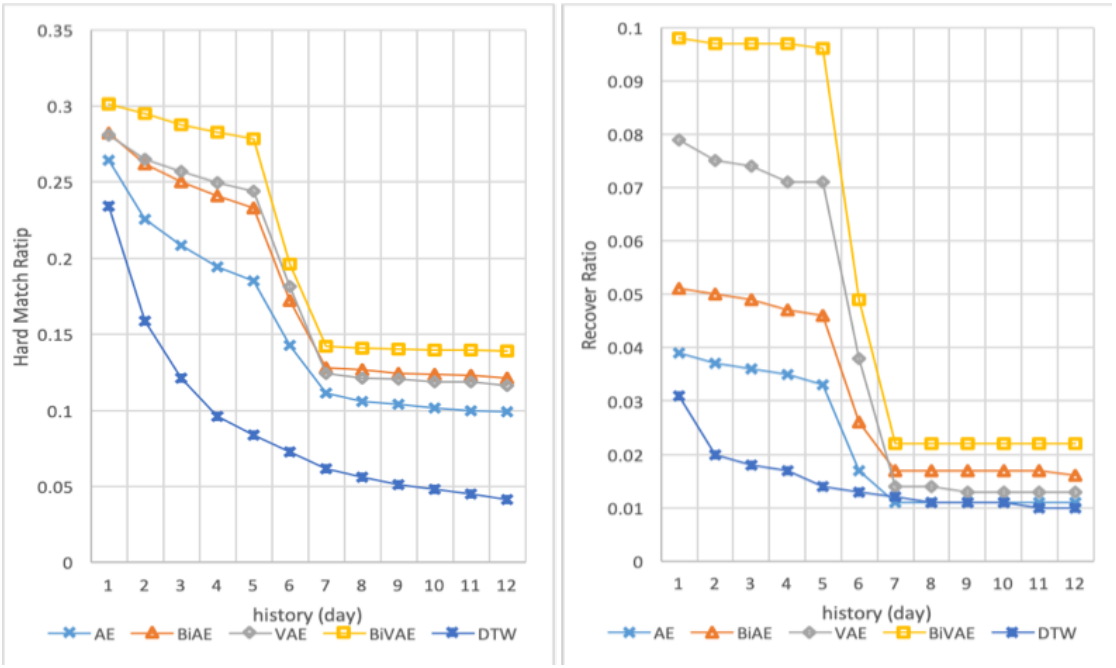


FIGURE 3.9: Comparison between different methods using HardMatchRatio and RecoverRatio.

Table 3.2: Comparison of different methods.

	DTW	EDR	EDwP	LCSS	AE	BiAE	VAE	BiVAE
CityEMD	-	17.1728	16.4438	17.1059	16.1798	16.1771	16.0617	16.0941
HMR	0.0615	0	0.0580	0.0005	0.1115	0.1280	0.1240	0.1420
RRatio	0.0120	0	0.0020	0	0.0110	0.0170	0.0140	0.0220

3.5 Conclusion

Retrieval based trajectory generator indeed solved some limitations of a VAE generative model for generating urban trajectories. The reconstructed trajectories come from historical dataset by latent index encoded by a trained generative model. While we also notice some limitations of current retrieval-based trajectory generator. The first one is that we need a large historical trajectory dataset to achieve diversity of reconstructed trajectory dataset, which is not always possible in reality. The second one is that we are not able to determine the scaling factor for each query trajectory by setting a distance threshold since the historical dataset itself could be biased, so the retrieved large trajectory dataset will also be biased. Therefore, our next step is to reduce the bias of trajectory dataset by incorporating information from other data sources such as census data.

Chapter 4

Differentiable Projection For Constrained Deep Learning

4.1 Introduction

Deep neural network (DNN) models have been widely used to solve many tasks in different domains and easily outperform traditional methods. The reason why people try to use the neural network model to replace the existing traditional methods, either because the neural network brings better accuracy, or because the neural network model saves a lot of time for calculations. For example, there are many existing studies devoted to solving various complex optimization problems using neural network models [58–60]. In the field of computer vision or natural language processing, it is more common to use more specific and efficient neural networks such as a convolutional neural network (CNN) [61–63] or recurrent neural network (RNN) [47, 55, 64] and its variant models. It has become a consensus for solving various complex problems, and have achieved better and more powerful performance compared to traditional methods.

However, above mentioned models, if trained under the framework of supervised learning or weak-supervised learning, they usually learned how to simulate the ground-truth data through the backpropagation of error between the network output and the ground-truth observation. In this process, the neural network as a "black box" is often criticized that the learning process of the network itself is a process of simulating the truth value, rather than understanding some of the basic rules of the ground-truth value that they should

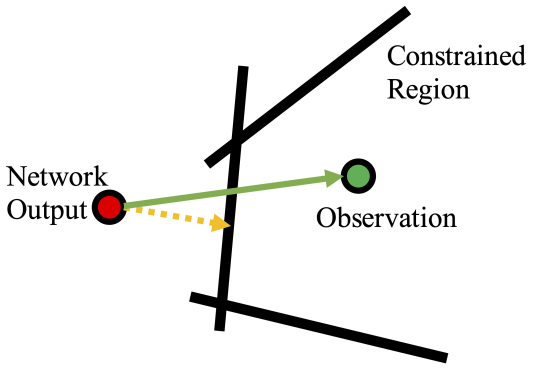


FIGURE 4.1: Ground truth observation (green point) belongs to a constrained region which formed by some linear constraints. Green line indicates the process of optimizing a conventional DNN. We consider the constrained region as a kind of prior knowledge, which should be incorporated in a conventional DNN (yellow dash line).

have learned [65]. Here we consider some application scenarios such as shown figure 4.1. In this scenario, we have ground truth observation included in constrained region, which formed by some linear constraints. Since it is known that all inferenced output of DNN should also belongs to the constrained region, we can consider those constraints as a kind of prior knowledge. For a conventional DNN model, the training process is directly minimizing the error between ground truth observation and network output, but an additional process should be optimizing the network output by constrained region, indicated by yellow dash line in figure 4.1. By combining these two process, the model is expected to be able to give some outputs closer to the constrained region, that will make the output closer to the ground truth observation as well. Neural network models incorporating with constraints are also used in solving optimization problems [66, 67]. There are also some novel neural network model for solving variational inequalities with linear and nonlinear constraints [68]. In addition, there are also many people who optimize the neural network model by adjusting the internal structure of the neural network to meet certain constraints [69, 70]. In recent years, we have observed a lot of work on constrained neural network models in the field of computer vision, such as a constrained convolutional layer to accomplish the problem that state-of-the-art approaches cannot detect resampling in re-compressed images initially compressed with high quality factor [71], and a constrained neural network (CCNN) which uses a novel loss function to optimize for any set of linear constraints on the output space of a CNN [72, 73]. Many of the previous studies on constrained neural network models are mostly dedicated to solving specific problems in one particular field, and there is a lack of a more general description of the problem. Besides, directly combing the DNN models

with constrained optimization algorithms, such as aforementioned CCNN, is not able to handle some more generalized constraints and is not time-efficiency since it is necessary to solve optimization problems for every batch network outputs. In this paper, we try to give a more unified problem formulation and use a general framework to solve the problem that the neural network output satisfy certain constraints. We hope that the output of the neural network model of supervised or weakly supervised learning is no longer a simple process of simulating the ground-truth value, but meets certain constraints. At the same time, we believe that when the output of the neural network always satisfies some of the constraint rules, the interpretability and transferability of the network can be improved. So here we try to describe this problem in detail as much as possible and show the feasibility of a general method to solve this problem.

4.2 Deep learning model with constraints

Given a dataset D that consists of input points $X = \{x_i \in \mathbb{R}^n, i = 1, 2, ..., N\}$ and its corresponding ground truth output points $Y = \{y_i | y_i \in \mathbb{R}^m, f_j(y_i) = b_j, g_k(y_i) \leq d_k, i = 1, 2, ..., N, j = 1, 2, ..., M, k = 1, 2, ..., T\}$, in which we assume that ground truth output Y satisfy some equality and inequality constrainst conditions. Equality constraints $f_j(\cdot)$ and inequality constraints $g_k(\cdot)$ are usually given by some physical or expert knowledge, which define a feasible set of possible inferenced outputs. In this research, we only focus on the linear constraints, then aforementioned constraint conditions can be rewritten as $\{f_i(Y) =< a_j, Y >= b_j, g_k(Y) =< c_k, Y >\leq d_k, j = 1, 2, ..., M, k = 1, 2, ..., T\}$, where $a_j, c_k, \in \mathbb{R}^m, b_j, d_k \in \mathbb{R}$, and $<, >$ is the Euclidean inner product.

Given a deep learning model, denoted by $N_\theta(\cdot)$, parameterized by θ, we use $Y^\star = N_\theta(X)$ denotes the output of this deep learning model. Our objective is to solve the following equation:

$$\begin{aligned} &\min_\theta (\mathbf{Y}^\star - \mathbf{Y})^2 \\ s.t. &f_i(\mathbf{Y}) =< a_j, \mathbf{Y}^\star >= b_j, j = 1, 2, ..., M \\ &g_k(\mathbf{Y}) =< c_k, \mathbf{Y}^\star >\leq d_k, k = 1, 2, ..., T \end{aligned} \tag{4.1}$$

For a conventional neural network, the outputs of the model is usually regularized by some nonlinear activation function. However, these widely-used activation is designed mainly for introducing nonlinearity in the neural network to increase the capability of

the model for approximating some complex mapping from inputs to outputs. These activation function used in neural networks lack the ability to ensure a feasible outputs of the model to satisfy the inequality constraints defined in Eq. 4.1. However, it is obvious that if the error between inferenced output of the deep learning model and the ground truth is small enough, the inferenced output should eventually satisfy the constraint conditions. Therefore, we consider the minimization of error as the ultimate goal, while the constraint conditions as a kind of prior knowledge, which could help us solve the minimization problem if utilized properly. The intuition of this idea is that if we can find a differentiable map denoted by $h : Y^\star \rightarrow \hat{Y}^\star$, where $\hat{Y}^\star$ is closer to all points in feasible region defined by the constraint conditions than original output $\mathbf{Y}^\star$, then we can obtain better inferenced output every time for sure comparing with the output of a conventional deep learning model. Therefore, we aim to propose a more generalized differentiable map which is designed properly to help solve Eq. 4.1.

4.2.1 Preliminary

Inspired by Pathak el at [72], we can define the map function $h : Y^\star \rightarrow \hat{Y}^\star$ as an implicit function because our goal is to obtain $\hat{Y}^\star$ and the implicit function h can be solved by following equations:

$$\begin{aligned} &\min_{Y^\star} D_{KL}(Y^\star || \hat{Y}^\star) \\ s.t. &f_i(\hat{Y}^\star) =< a_j, \hat{Y}^\star >= b_j, j = 1, 2, ..., M \\ &g_k(\hat{Y}^\star) =< c_k, \hat{Y}^\star >\leq d_k, k = 1, 2, ..., T \end{aligned} \tag{4.2}$$

To solve Eq. 4.2, we could use Lagrange multiplier

$$\begin{aligned} &L(\hat{Y}^\star, \lambda, \mu) = D_{KL}(Y^\star || \hat{Y}^\star) + \sum_{j=1}^{M} \lambda_j f_j(\hat{Y}^\star) + \sum_{k=1}^{T} \mu_k g_k(\hat{Y}^\star) \\ &\lambda_j f_j(\hat{Y}^\star) = 0, f_j(\hat{Y}^\star) = 0 \\ &\mu_k g_k(\hat{Y}^\star) = 0, g_k(\hat{Y}^\star) \leq 0 \\ &\mu_k \geq 0, j = 1, 2, ..., M, k = 1, 2, ..., T \end{aligned} \tag{4.3}$$

The optimal point $\hat{Y}^\star$ of an analytic solutions is obtained when

$$\frac{\partial}{\partial \hat{Y}^\star} L(\hat{Y}^\star, \lambda, \mu) = \frac{\partial}{\partial \hat{Y}^\star} D_{KL}(Y^\star || \hat{Y}^\star) + \sum_{j=1}^{M} \lambda_j \frac{\partial}{\partial \hat{Y}^\star} f_j(\hat{Y}^\star) + \sum_{k=1}^{T} \mu_k \frac{\partial}{\partial \hat{Y}^\star} g_k(\hat{Y}^\star) = 0 \quad (4.4)$$

Here we only consider the case that the output of the deep learning model $Y^\star$ doesn't satisfy the linear constraint conditions. It is because that all constraint will not be active when the output $Y^\star$ satisfy the constriant conditions, then the closest point $\hat{Y}^\star$ is the same with output $Y^\star$ exactly. From the definition of the point $\hat{Y}^\star$, it is obvious that all $\hat{Y}^\star$ should located in the feasible region of the linear constraint condition. At the same time, since an optimal $\hat{Y}^\star$ should be the closest with $Y^\star$, we know that $\hat{Y}^\star$ should be on the boundary of the feasible region, and the line connected by $Y^\star$ and $\hat{Y}^\star$ is orthogonal with hyperplane where $\hat{Y}^\star$ located on since $\frac{\partial}{\partial \hat{Y}^\star} \mathrm{L}(\hat{Y}^\star, \lambda, \mu) = 0$. Althogh we don't know the explicit function h, we can obtain an identical point $\hat{Y}^\star$ by solving aforementioned equations.

So far, we have given the solution for solving Eq. 4.2, by solving this KKT conditions, we are able to find the optimal results, which is the closest points of original network outputs in the constrained region. However, directly solving this constrained optimization problem is not a efficient solution which requires heavy computation since it is computed every iteration when the DNN produces some outputs. As discussed in previous section, constrained convolutional neural network [72] is a kind of method which solves KKT conditions every time when training. We doubt this strategy could be used in more generalized scenarios. Actually, we also think that finding the closet point $\hat{Y}$ of original network output in the constrained region is not necessary for solving Eq. 4.3. That is because what we want to achieve is that the output of DNN should be close to the constrained region. In the best situation, the output of DNN should satisfy the constraint. So directly solving KKT conditions for guiding the training process of DNN model is waste of computational resource. Therefore, we aim to find other approach to solve the Eq. 4.3.

4.2.2 Projection methods for linear equalities

Let us consider a specialized scenario that there is only linear equalities constraints for the output of neural network. We can rewrite Eq. 4.2 into matrix form like:

$$AY^{\star} = b \tag{4.5}$$

Where $A \in \mathbb{R}^{k,m}, b \in \mathbb{R}^{k}$. We can easily obtain the condition for $Y^{\star}$ have feasible solutions is that $rank(A) \leq k \leq m$. It means that there exists a feasible area when the number of efficient constraints a_j is less than the dimensionality of the variable $Y^{\star}$. More specially, if the rank of matrix A satisfy that $rank(A) = k$, we have following lemma:

Lemma 1. Given any $A \in \mathbb{R}^{m,n}$, and $rank(A) = m \leq n, b \in \mathbb{R}^{m}$. the closest point $x^{\star}$ subject to $Ax = b$ is $x^{\star} = (I - A^{T}(AA^{T})^{-1}A)x + A^{T}(AA^{T})^{-1}b$. The proof of Lemma 1 is given in Appendix. Based on Lemma 1, given any network output $Y^{\star}$, we can find the closest point in feasible area $\hat{Y} = (I - A^{T}(AA^{T})^{-1}A)Y^{\star} + A^{T}(AA^{T})^{-1}b$. Then we define the loss function of the DNN model to be $l = \alpha(Y - Y^{\star})^2 + (1-\alpha)(Y - \hat{Y})^2$. From figure 4.2, we could see that if the first term of loss function is not zero, the second term of loss function will be zero if the original output of neural network is in the normal direction of hyperplane of ground truth point, which means the projected output will be exact ground truth point. So we could know that the second term of loss function encourages the neural network give some predicted points that is in the normal direction of hyperplane of their ground truth points.

We could also know that if the original output of neural network directly satisfy the equalities constraints, which shown by $Y^{\star} = \hat{Y} = (I - A^{T}(AA^{T})^{-1}A)Y^{\star} + A^{T}(AA^{T})^{-1}b$. We could see the loss function become just the widely used mean square error in most networks between ground truth point and original output of neural network: $l = (Y - Y^{\star})^2$. In this condition, there will be no additional penalty of loss and the structure of the projection matrix didn't activated, the model is a pure conventional neural network.While for a more generalized scenario, the constraints is not always independent with each other, but also there could exist some inequalities constraints in many applications.

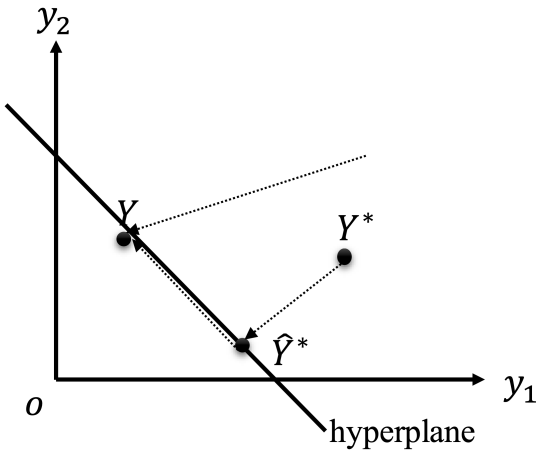


FIGURE 4.2: loss function contains two terms, the first term penalize the output of neural network to be close with ground truth, while the second term will be large if the output of neural network is not in normal direction of hyperplane of ground truth.

4.2.3 Projection methods for linear inequalities

While for a more generalized scenario, the constraints is not always independent with each other, but also there could exist some inequalities constraints in many applications. This section, we will give a projection method for linear inequalities constraints problems. Without any assumption about linear independent of constraints of Eq. 4.2, we define a partitional projection like:

$$h_k = \min\{0, \frac{d_k - < c_k, Y^\star >}{c_k, c_k}\} \tag{4.6}$$

Then:

$$P_k Y^\star = Y^\star + h_k(Y^\star)c_k \tag{4.7}$$

Above equation defines the orthogonal projection of the point $Y^\star$ on the closed half space:

$$H_k = \{Y^\star \in \mathbb{R}^m :< c_k, Y^\star > \leq d_k\} \tag{4.8}$$

Now, let us choose some real numbers satisfy:

$$\sum_{k=1}^{T} \lambda_k = 1, \lambda_k > 0 \tag{4.9}$$

Then, we can define $P : \mathbb{R}^m \to \mathbb{R}^m$ by:

$$P = \sum_{k=1}^{T} \lambda_k P_k \tag{4.10}$$

For any $\alpha \in \mathbb{R}$,

$$P_\alpha = I - \alpha(I - P) \tag{4.11}$$

Where I is the identity matrix. with above definition, we can use Eq. 4.11 to find a feasible solution of $\hat{Y}$ which satisfy constraints of Eq. 4.2, given any output $Y^\star$ of deep learning models. Given an output $Y^\star$ of deep learning models, define inductively

$$\begin{aligned} Y^0 &= Y^\star \\ Y^{t+1} &= P_\alpha Y^t \\ Y^{t+1} &\to \hat{Y}.t \to \infty \end{aligned} \tag{4.12}$$

With $0 < \alpha < 2$. The detailed proof of the convergence of above algorithm is given in [74]. Given an initial point $Y^0 \in \mathbb{R}^m$, let $\{Y^t\}$ be the sequence defined by Eq. 4.11. There are two elementary properties of orthogonal projections from theorem 11.2 in [75]. For any $x, y \in \mathbb{R}^n, 1 \leq i \leq m$

$$< P_i y - P_i x, x - P_i x > \leq 0 \tag{4.13}$$

$$||P_i y - P_i x|| \leq ||y - x|| \tag{4.14}$$

From Eq. 4.13, we know that for any i,

$$\begin{aligned} &< P_i y - P_i x, x - P_i x > \leq 0 \\ \Rightarrow &< P_i y - P_i x, x > \leq < P_i y - P_i x, P_i x > \\ \Rightarrow &< Py - Px, x > \leq \sum_{i=1}^{n} \lambda_i < P_i y - P_i x, P_i x >. \end{aligned} \tag{4.15}$$

If we substract $< Py - Px, Px >$ from above inequality, we will have:

$$\begin{aligned}
&< Py - Px, x - Px > \leq \sum_{i=1}^{n} \lambda_i < P_i y - P_i x, P_i x > - < Py - Px, Px > \\
&= (\sum_{j=1}^{n} \lambda_j) \sum_{i=1}^{n} \lambda_i < P_i y - P_i x, P_i x > - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j < P_i y - P_i x, P_j x > \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j < P_i y - P_i x, P_i x - P_j x > \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j (< P_i x - P_j x, P_i y - P_j y > - ||P_i x - P_j x||^2) \\
&\leq \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j (||P_i x - P_j x|| ||P_i y - P_j y|| - ||P_i x - P_j x||^2)
\end{aligned} \tag{4.16}$$

Lemma 2. Let $F = \{z \in \mathbb{R}^m : Pz = z\}$, for any $z \in F, Y \in \mathbb{R}^m, < z - PY, Y - PY > \leq 0$.

Proof.

$$\begin{aligned}
&< z - PY, Y - PY > = < Pz - PY, Y - PY > \\
&\leq \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j (||P_i Y - P_j Y|| ||P_i z - P_j z|| - ||P_i Y - P_j Y||^2)
\end{aligned} \tag{4.17}$$

$$\begin{aligned}
&0 = < PY - Pz, z - Pz > \\
&\leq \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j (||P_i z - P_j z|| ||P_i Y - P_j Y|| - ||P_i z - P_j z||^2)
\end{aligned} \tag{4.18}$$

Add Eq. 4.17 and Eq. 4.18 togather:

$$< z - PY, Y - PY > \leq \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j (||P_i Y - P_j Y|| - ||P_i z - P_j z||)^2 \leq 0 \tag{4.19}$$

■

Lemma 3. For any $z \in F, Y^0 \in \mathbb{R}^m, ||Y^t - z||$ decrease if $F \neq \Phi$.

Proof.

$$
\begin{aligned}
&||Y^{t+1} - z||^2 = ||P_\alpha Y^t - z||^2 = ||(I - \alpha(I - P))Y^t - z||^2 \\
&= ||Y^t - z + \alpha(PY^t - Y^2)||^2 \\
&= ||Y^t - z||^2 + ||\alpha(PY^t - Y^t)||^2 + 2 < Y^t - z, \alpha(PY^t - Y^t) > \\
&= ||Y^t - z||^2 + \alpha(\alpha - 2)||(PY^t - Y^t)||^2 + 2 < Y^t - z, \alpha(PY^t - Y^t) > +2\alpha||PY^t - Y^t||^2 \\
&= ||Y^t - z||^2 + \alpha(\alpha - 2)||PY^t - Y^t||^2 + 2\alpha < PY^t - z, PY^t - Y^t >
\end{aligned}
\tag{4.20}
$$

The second term of Eq. 4.20 is negative since $\alpha \in (0, 2)$, and the third term is nonnegative by Lemma 2.

$$||Y^{t+1} - z||^2 \leq ||Y^t - z||^2$$

■

Define the positive function $f : \mathbb{R}^m \to \mathbb{R}$:

$$f(Y) = \sum_{j=1}^{k} \lambda_j ||P_j Y - Y||^2 \tag{4.21}$$

Theorem 3. For any starting point $Y^0 \in \mathbb{R}^n$, the sequence $\{Y^k\}$ generated by Eq. 4.11 converges. If the system of Eq. 4.2 is consistent, the limit point is a feasible point for Eq. 4.1. Otherwise, the limit point minimizes $f(Y) = \sum_{j=1}^{k} \lambda_j ||P_j Y - Y||^2$, i.e., it is a weighted (with the λ_i's) least squares solution to Eq. 4.2.

Detailed proof of Theorem 3 is given in Appendix. Here we modify above algorithm to get a differentiable approximated projection method. We know that if we could eventually find a projected output $\hat{Y}$ in feasible region given the Network output $Y^\star$, but this process could be time-consuming potentially since many projection steps might be necessary. But fortunately, there is a good property for this projection method given by Lemma 3, which indicates that projected points at current step are always closer to the constrained region than those projected points at last step. Therefore, only limited steps of projection is needed, and the projected outputs are always better than original network outputs. We give a simplified projection method as:

$$\hat{Y}_t = Y^\star - \sum_{k=1}^{T} H(C_k Y^\star) P_k Y^\star - \sum_{j=1}^{M} P_j Y^\star \tag{4.22}$$

Where $Y^\star$ is the original deep learning model outputs, $H(x) = \frac{d}{dx}\max(x, 0)$ is the unit step function, P is the projection matrix $P = A^T(AA^T)^{-1}A$. Eq. 4.22 forms a differentiable projection layer. Since we aim to solve Eq. 4.3 which including equality and inequality constraints, Algorithm 1 is proposed. We notice that there is no assumption about the linear independent of constraints, which means that the rank of A is not always to be full row rank, so we summarize above equations in a more uniform way.

Algorithm 1: Differentiable projection method for linear inequalities

Input: $X = \{x_i \in \mathbb{R}^n, i = 1, 2, ..., N\}, a_j, b_j, c_k, d_k$, which define the equality and inequality constraints. α which is the hyperparameter for loss function. Randomly initialize parameters $\lambda_j, \lambda_k, s.t. \sum_{j=1}^{M} \lambda_j + \sum_{k=1}^{T} \lambda_k = 1, \lambda_j > 0, \lambda_k > 0.$;

Output: $Y = \{y_i \in \mathbb{R}^m, i = 1, 2, ..., N\}$;

Initialize a deep learning model with parameters θ denoted as $N_\theta(\cdot), epoch = 1, MaxEpochs$;

while $epoch \leq MaxEpochs$ **do**

$Y^\star = N_\theta(X)$;

$\hat{Y}_t = Y^\star - \sum_{k=1}^{T} H(C_k Y^\star) P_k Y^\star - \sum_{j=1}^{M} P_j Y^\star$;

$loss = \alpha(Y - Y^\star)^2 + (1 - \alpha)(Y - Y^0)^2$;

$loss.backward()$;

$epoch += 1$;

end

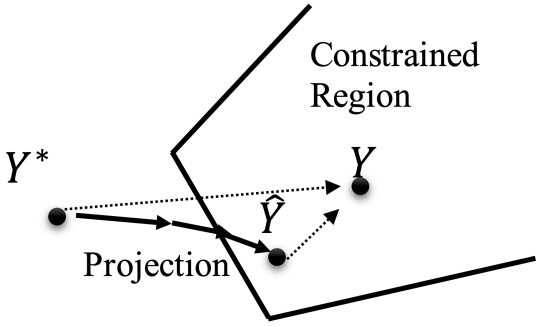


FIGURE 4.3: the original output of networks is projected to be a feasible point in constrained region.

Figure 4.3 shows the process when training a neural networks with inequalities constraints and equalities constraints. At each time, the networks output some original points $Y^\star$, by using projection method defined in algorithm, there will be some converged feasible points $\hat{Y}$ located in constrained region. We have to clarify that the projected feasible points $\hat{Y}$ is not always the KKT points of original output $Y^\star$, which

means that projected feasible points is not always the closest points of original output. By using projection method, we is just find some possible feasible solutions but not the best solution. Fortunately, from the definition of loss function, we notice that this method still work.

After adding an projection module, we force the final output of the model satisfy the constraints. Then, we calculate errors using both adjusted output $\hat{Y}$ and original output $Y^\star$, and backpropagate both errors to the neural network for training. The advantages of this model structure are two-folds: 1) the final output $\hat{Y}$ after the projection module is forced to satisfy the constraints; 2) the model follows the direction of letting both original prediction $Y^\star$ and adjusted prediction $\hat{Y}$ to be close to the ground truth Y as much as possible.

Figure 4.3 is given for a more detailed explanation for the advantage of adding an projection module. It is obvious that if we want our projected output of neural network satisfy the constraints, we should select some candidates only from the constrained region instead of the entire $\mathbb{R}^M$ space. While, the original output $Y^\star$ is defined in the $\mathbb{R}^M$ space, so we want the adjusted output $\hat{Y}$ to be close to the original output of neural network as well. When $Y^\star$ is in the feasible area, the projection module will be not activated. Then we have $Y^\star = \hat{Y}$, and the loss function become MSE which is the same with projection method defined in linear equalities constraints.

4.3 Numerical Experiments

In this section, we present some numerical experiments to demonstrate the effectiveness of the constrained neural network. The data we used for numerical experiments are randomly generated synthetic data. In this experiment, we use exactly same parameter settings and training process for a fair comparison with different methods. The input data is generated by a 64-dimensional unit Gaussian distribution. Then we build a nonlinear function with random parameters to convert the 64-dimensional inputs to a 8-dimensional vectors. The linear equality constraints and inequality constraints are all generated randomly, and we only keep the data pair which satisfy this random constraints. The objective of this problem is to evaluate the performance of projection DNN using MSE as metric.

4.3.1 effectiveness of projection layer

In this experiment, we compare the projection DNN (PDNN) with a conventional DNN. We also compare the results of conducting projection as post processing of DNN, denoted as "DNN+projection". It can be found that the PDNN gives the best results, while DNN+projection also outperform the conventional DNN. That prove our assumption that by incorporating some linear constraints as prior information, even a simple model could give better results.

TABLE 4.1: Comparison of results.

	DNN	DNN + Projection	PDNN
MSE	0.02279	0.02268	**0.02093**

4.3.2 Ablation

We also evaluate the impact of the number of projection layer and the hyperparameter α in the loss function. For a single iteration, from Lemma 3, we know that more projection layers in a PDNN model, the projected outputs will be closer to the ground truth observations. This will lead to an intuition that we should use more projection layer in a PDNN to get a better performance. While the experimental results deny this kind of intuition shown in Figure 4.4 (left). At least in this experiment settings, the best performance is obtained when the PDNN consists of 3 stacked projection layer. The number of projection layer in a PDNN to achieve the best performance could be different in another application scenarios. Still, based on this results, it suggests that using limited stack projection layers could be sufficient instead of stack as many projection as possible which will lead to increased computational burden.

Since the loss function is originally defined as $l = (1-\alpha)(Y - Y^\star)^2 + \alpha(Y - \hat{Y})^2$, we also evaluate the impact of the hyperparameter α. The best performance is obtained when $\alpha = 1$ shown in Figure 4 (right). This result suggest that we should only use projected outputs for calculating the loss to guide the training of the PDNN. However, we think it depends on the different scenarios to decide the hyperparameter α.

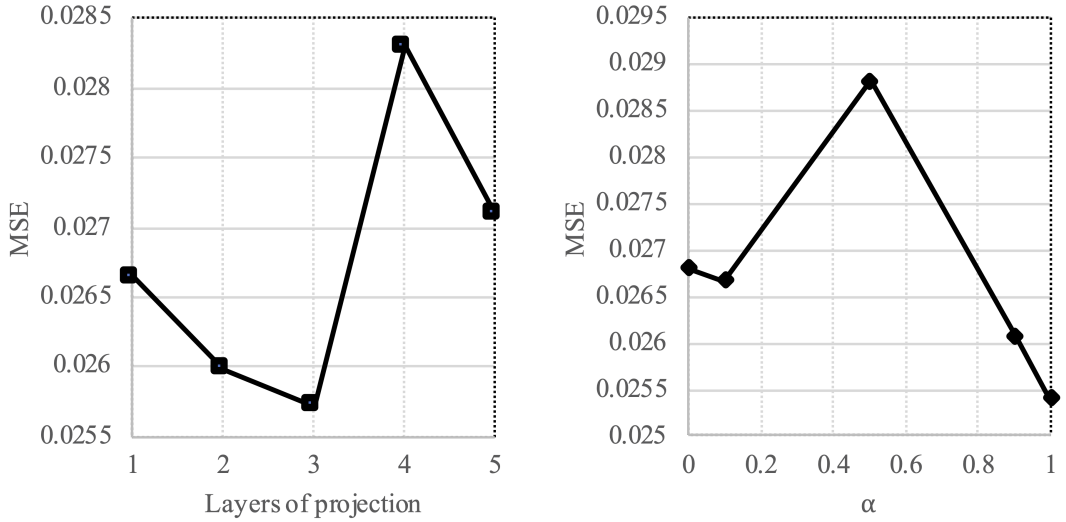


FIGURE 4.4: The performance when using different number of projection layer (left); the impact of the hyperparameter α in the loss function (right).

4.3.3 Impact of constraints

Experiments are also conducted for the comparison of different methods to evaluate the performance under different constraints conditions. The loss function of "DNN+resid" in table 4.2 is $loss = (Y^\star - Y)^2 + \max(AY^\star, 0)$. When the network output $Y^\star$ didn't satisfy the constraints, $AY^\star$ will be a positive value, which is greater than 0. If the network output satisfy the constraints, then the residual will be 0. The loss function of "DNN+fix resid" is similar, when the network outputs didn't satisfy the constraints, give a fix penalty in the loss function.

TABLE 4.2: The performance of different methods under different constraints conditions.

constraints	DNN	DNN+fix resid	DNN+resid	DNN+Proj	PDNN
3 * 8	0.08979	0.05991	0.06106	0.08979	0.09090
4 * 8	0.09952	0.04378	0.04472	0.09951	0.09872
5 * 8	0.12691	0.04946	0.05587	0.09055	0.07386
6 * 8	0.18655	0.26512	0.26483	0.18655	0.18650
7 * 8	0.10872	0.13634	0.13788	0.10871	0.10817

From table 4.2, we notice that in most case, the PDNN outperforms conventional DNN and DNN+projection. we must also figure out that the baseline methods "DNN+resid" and "DNN+resid" sometimes give significantly better results than other 3 methods, while they are not robust. When the constraint conditions change, the performance

of these 2 methods become worth than other 3 methods. The possible reason for this phenomena could be that when the complexity of constraint condition increases, residual of the outputs will increase dramatically, then the model is not able to give accurate predictions. This might be solved by tuning the hyperparameters in the model, but it requires more attempt efforts than PDNN or conventional DNN. so another advantage of the PDNN is that we could always get better results without additional hyperparameter tuning efforts.

4.4 Conclusion

In this chapter, we aims to solve problems with linear constraints as a kind of prior knowledge in a more generalized form. We discuss the necessity of solving this kind of constrained problems under KKT conditions which is widely used before. Also, we show that in linear constrained case, we could use a projection method to build a differentiable projection layer of the DNN efficiently. We use the dot product of errors of both original output of the DNN and projected output to train the DNN. Besides, when the original output is in the feasible region, we show that conventional DNN is a special case of this projection DNN. Then, we conduct numerical experiments using randomly generated synthetical dataset to evaluate the performance of the PDNN including comparison with a convention DNN and some different simple modification. The experimental results show the effectiveness and robustness of the PDNN. Currently, the projection layer proposed is differentiable while there is only fix parameters. In the future, we will investigate the improvement of this projection layer to achieve a projection layer structure which consists of learnable parameters. Also, we will evaluate our methods for some real world applications.

Chapter 5

Linear constrained human trajectory generation

5.1 Introduction

The task of estimating the movement trajectory of a person can usually be divided into two subtasks: 1) complete the model of the reconstruction of the personal trajectory; 2) complete the estimation of the scaling factor by combining heterogeneous information within the city. We hope to construct a constrained generative model as a constrained trajectory generator to solve the two subtasks simultaneously.

5.1.1 Problem definition

We usually only get some relatively small and low sampling rate biased data. Here, we first define the meaning of low sampling rate and sampling bias in this chapter:

1) The low sampling rate means that the number of unique user IDs in human mobility data collected is usually not big enough to represent the real-world population. For example, the obtained data set only accounts for less than 1% of the actual population.

2) The sampling bias is due to the significant differences in the groups targeted by the human mobility data obtained by different data collection techniques, resulting in the data obtained generally focusing and showing the movement behavior patterns of

a particular group of people. For example, the human mobility GPS dataset collected by relying on the navigation system installed in the car shows the mobility pattern of the car. This mobility pattern is quite different from the mobility of trains, bicycles, or walking people. We define the difference between the representativeness of the data collected from a particular group of people and the human mobility pattern of all people in the actual scene as bias.

Another issue is that we also need to consider how to prevent privacy violations when using human mobility data. Because the human mobility GPS data we use contains a lot of detailed user movement information, such as the user's home address, work address, and frequently visited Point of Interest (POI). Users usually have a strong sensitivity to whether this type of data will infringe on their privacy. At the same time, the service provider also needs to comply with ethical and even legal provisions to protect privacy and even encrypt the collected data. Therefore, even the maturity of various data-driven algorithms, such as deep learning, brings new opportunities to solve many problems in cities based on human mobility data that were difficult to solve before, such as traffic congestion, the impact of large-scale events on human mobility patterns et al., the privacy violation problem remains. The basis of these data-driven algorithms is to use as much human mobility data generated in the real world as possible, so the question becomes how to make full use of helpful information contained in human mobility data without violating users' privacy.

5.1.2 Research objective

Our objective for building a constrained trajectory generator is to scale up a biased small trajectory dataset given some information from other data sources. Our expected outputs should be a less biased large trajectory dataset. At the same time, the model we proposed should solve the two issues mentioned above.

5.2 Methodology

5.2.1 Preliminary

In the previous chapters, we introduced a retrieval-based approach for human mobility generation that uses the similarity compressed to the hidden Gaussian mixture distribution as the human mobility trajectory similarity and retrieves historical human mobility data with a higher degree of similarity to tackle the problem of the low sampling rate in collected human mobility data. The retrieval-based human mobility generator proposed before is mainly to solve the problem that it is difficult for us to measure the authenticity of the trajectory generated directly by the generative model. However, in the scenario we anticipated, we cannot get a large amount of historical human mobility data because this runs counter to the premise that we need to protect user privacy, which means that we cannot use the retrieval-based human mobility generator proposed before to generate many trajectories close to reality.

Generative model One of the reasons we use generative models is because we want to have more diversity when generating virtual human mobility GPS trajectories. Here, the definition of diversity is that many users have similar movement patterns, but there are still some minor differences in the detailed trajectories they generate in their daily activities. We define this subtle difference as the diversity of trajectories. Previously, the retrieval-based human mobility generator was used based on the similarity of the trajectory based on the individual mobility pattern represented by the compressed hidden vector. Then trajectories similar to the current query trajectory in the historical GPS trajectory database were retrieved as the virtual generated GPS trajectory. The advantage of this method is that every selected GPS trajectory has been recorded in history, so we do not need to measure the authenticity of the generated virtual trajectory. Its main limitations are in two aspects. 1) We need an extensive historical data set to construct a rich and diverse GPS trajectory set. Otherwise, we will not be able to guarantee that the new virtual data generated has diversity and a certain degree of accuracy. 2) Using this method, we need a pre-calculated scaling factor for each query GPS trajectory to generate enough virtual GPS trajectory data. This generated virtual human mobility GPS trajectory data needs to be close to the citywide human mobility pattern in the real world.

Correspondingly, the disadvantage of directly using the generative model is that the generated trajectory often does not perfectly conform to the topological structure of the urban road network. Therefore, when we directly use the generative model to generate GPS trajectories, post-processing such as map matching is usually required. However, directly using the generative model to generate GPS trajectories also brings some benefits, such as 1) We do not need to maintain a vast historical database, which aligns with our goal to protect users' privacy while generating more virtual human mobility GPS trajectories. 2) We can directly generate diversity by controlling the Gaussian mixture distribution generated by each GPS trajectory in the generative model. Every time a new virtual GPS trajectory is generated, we need to perform random sampling on a Gaussian mixture distribution. Next, we will briefly introduce how to use the generative model to generate a new virtual GPS trajectory.

We usually use an Encoder-Decoder structure to construct our generative model. When we follow the variational inference framework, the Encoder refers to map the GPS trajectory from an input individual to a multivariate Gaussian mixture distribution. In contrast, the decoder refers to map a random point we are sampling from a particular multivariate Gaussian mixture distribution to a GPS track of an individual. The core of this model is the structure of multivariate Gaussian mixture distribution, which is the key to generating new virtual GPS trajectories. When we map our current low-sampling rate human mobility data with certain deviations into many multivariate Gaussian mixture distributions through a generative model, we only need to use specific sampling methods to generate more virtual human mobility GPS data. We choose a proper sampling method to random sampling points in obtained multivariate Gaussian mixture distributions to generate the virtual trajectory data we finally want, which can reduce the deviation compared with the original trajectory data.

Heterogenous data as guide information We also need to combine some heterogeneous data to the human mobility GPS trajectory dataset, the most typical being statistical data. This chapter intends to use ubiquitous demographic data such as OD distribution as heterogeneous data to supplement some census information to guide scaling factors. One of our assumptions is that we down-sample the human mobility GPS trajectory dataset and only extract the OD information of each trip as demographic data. Then we grid this simulated OD table, and the data obtained after visualization

is shown in the figure 5.1. We hope that this OD distribution data can guide us in estimating a more reasonable scaling factor to reduce the bias generated when the virtual human mobility GPS trajectory data set is generated.

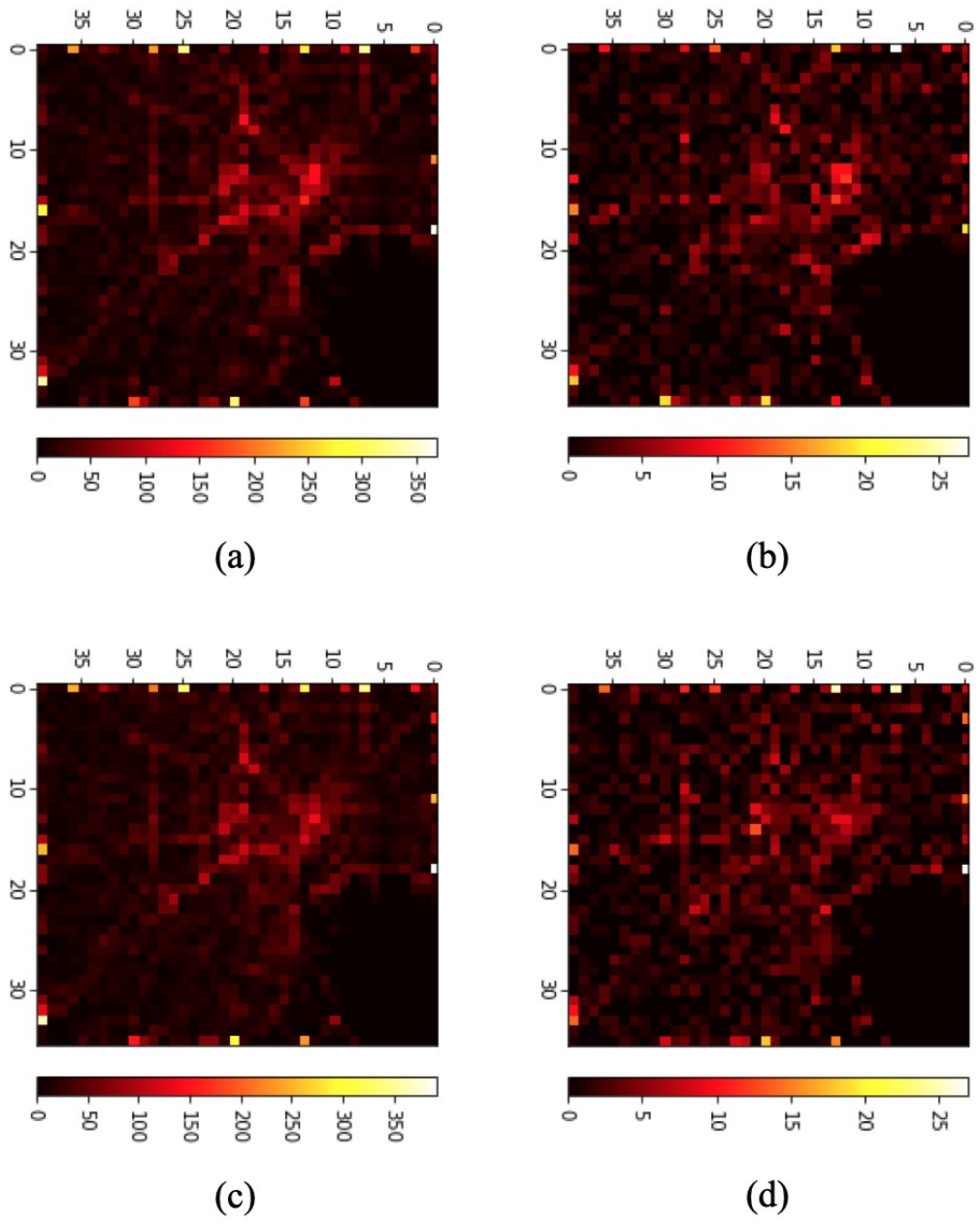


FIGURE 5.1: OD distribution of one day (left (a), (c), 2019.10.31) and one hour (right (b), (d).

In a perfect situation, if we can estimate an excellent scaling factor, denoted as w, it should satisfy the following equation:

$$OD_{hour} * w = OD_{estimate} \tag{5.1}$$

Therefore, we know that after we have established the relationship between the information contained in heterogeneous data and the scaling factor, our crucial question has become how to effectively use the guidance information to estimate an excellent scaling factor.

At the same time, we have also noticed that it is not correct to rely solely on OD distribution data to optimize the scaling factor. Every time we generate a new virtual human mobility GPS trajectory through a generative model, it is not the same as the real input GPS trajectory. Therefore, we believe that a more compact method is constructing a method that can update weighted multivariate Gaussian mixture distribution according to the scaling factor estimated each time. When we generate multiple multivariate Gaussian mixture distributions constructed by the model, randomly sampling enough random noise points on this complex distribution, then input these random noise points into the generating model to reconstruct a virtual human mobility GPS trajectory. Then we can know that the key to our construction of this model is how to construct the relationship between the information of the OD distribution and the information of the trajectory sampled from the estimated weighted multivariate Gaussian mixture distribution.

one-hot encoding for trajectory In machine learning, a one-hot is a group of bits of which the legal combinations of values are only those with a single high (1) bit and all the other low (0) [76]. We transform the original human mobility GPS trajectory data as shown in the below. We first transform the human mobility GPS trajectory from the original trajectory data, carrying timestamp, longitude, and latitude information, into cities with pre-divided grids according to the latitude and longitude information and obtain a grid composed of timestamp and grid ID. A simplified trajectory sequence composed of grid IDs, and since the ID of each grid in each city is unique, we can perform one-hot encoding on the grid ID at each moment and encode each grid ID into The corresponding one-hot vector. In this chapter, since we normalized the trajectory into a sequence with a fixed time interval, the fixed time interval is 1 minute. We can discard the extra dimension used to represent the moment of each point in the sequence. According to this rule, we only need to arrange the encoded one-hot vectors in chronological order to characterize the information contained in a human mobility trajectory completely. At this time, the trajectory is converted into a matrix composed of multiple one-hot vectors.

In a city with grids divided in advance, suppose we use the current number of people in each grid to represent a kind of "population density," denoted as $b \in \mathbb{R}^m$. Then, our scaling factor is w, where m is the city's grid number, n is the number of query GPS trajectories. In the above, we have converted the human mobility GPS trajectory into a matrix composed of one-hot vectors, and these one-hot vectors are arranged in chronological order. Then the origin of each human mobility GPS trajectory is also a one-hot vector. We use the following formula to express the $i-th$ trajectory:

$$A_i \in \mathbb{R}^m, \sum A_i = 1 \tag{5.2}$$

Then we know that since we want to use population density to estimate the scaling factor of each trajectory if we write query trajectory as $A \in \mathbb{R}^{m,b}$, we can get the following relationship:

$$Aw = b \tag{5.3}$$

This formula is established because we need the statistical information of the virtual human mobility GPS trajectory data set we generated to be consistent with the statistical data. For the destination of each trajectory, we can get a similar formula through the same principle. Therefore, the problem becomes that how can we guarantee that the generative model gives a $w, s.t. Aw = b$ every time?

Projection method In previous research, Pathak [72] proposed a two-step method to solve the training and learning of constrained deep learning networks and applied it to the image recognition problem in the field of computer vision. Their leading ideas are shown in the figure below. We observe that their method has two main limitations: 1) This method cannot be implemented for heterogeneous input data. 2) This method requires much computational cost. In order to avoid these two limitations of their method, we propose to use the projection method to solve the training and optimization problem of this constrained deep learning model. According to some conclusions in Chapter 4, we can know the following information:

For $A \in \mathbb{R}^{mxn}$, and $rank(A) = m$. Given any $w \in \mathbb{R}^n$, the closest point, denoted as $\hat{W}, s.t. A\hat{w} = b$ is

$$\begin{aligned} \hat{w} &= (I - A^T(AA^T)^{-1}A)(w - \hat{w}) + \hat{w} \\ &= (I - A^T(AA^T)^{-1}A)w + A^T(AA^T)^{-1}b \end{aligned} \tag{5.4}$$

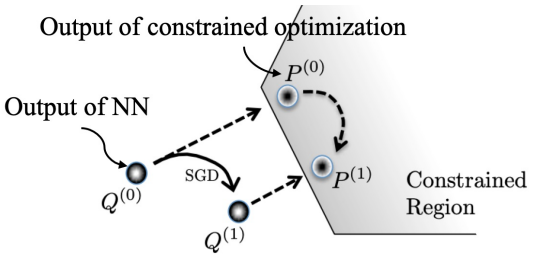


FIGURE 5.2: Two step approach for training a Deep learning model with constraints.

The above equation provides a straightforward projection method for solving a linear constrained optimization problem if the condition that $rank(A) = m$, which means the matrix A has full column rank. However, we found that we found that the one-hot matrix composed of query human mobility GPS trajectory is not full column rank in the real world. The reason is as follows.

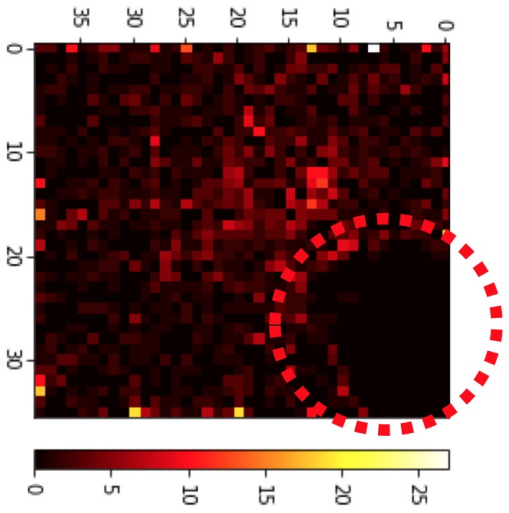


FIGURE 5.3: A geographical explanation for the matrix $rank(A) \neq m$. The red dash circle shows the sea, so there are no people lives here.

To solve the problem that $rank(A) \neq m$, we have two alternative solutions: 1) Padding. 2) Loosen constraints. The first solution is that we can add an additional matrix, denoted as $\hat{A}$, to make sure that $rank([A\hat{A}]) = m$. The limitation of this solution is that the scaling factor w estimated by this matrix is sometimes meaningless. The second solution is to loosen some linear constraints to make sure the loosened human mobility GPS one-hot matrix, denoted as A^{-}, satisfies the equation $rank(A^{-}) = rank(A)$. Then, we also need to find the corresponding loosen population density, denoted as $b^{-} = b[A^{-}.index]$. There is also a limitation that some constraints are lost due to this process. In the above

equation, A^- denotes the matrix in which some columns that are not independent with other columns in human mobility GPS trajectory one-hot matrix has been dropped, and b^- denotes that we drop the population density information corresponding to those dropped columns in matrix A.

5.2.2 framework

At first, we will explain how to build some constraints as a proper relationship between information from other data sources with every single trajectory. Suppose we can access some census data, such as the OD dataset. We used simulated OD data from the trajectory dataset in this research, but the framework of this method has no difference if we switch to a real OD dataset. The OD dataset gives us information on how many people move from a particular origin to a specific destination. While in the trajectory dataset, we only capture the movement trajectory of only a small number of people, which we call a biased trajectory dataset. Using previous approaches, we could also generate more trajectories based on existing trajectories, but these methods cannot reduce the bias. We need the information of the OD dataset to reduce the bias of the small trajectory dataset. In this research, we consider a simple constraint at first: the number of generated trajectories should be the same as the number in the OD dataset.

The first step is dividing the experimental area into grids, so the trajectory is converted from a sequence of longitude and latitude to a sequence of grid ID. Then we further convert this sequence of grid ID to a sequence of one-hot vector, so the trajectory becomes a matrix that each column contains only a "1" in the index of grid ID, with other elements all to be "0". We denote this matrix as $A \in \mathbb{R}^{(m,n)}$, where n is the number of trajectories, and m is the number of grid IDs. For an OD dataset, we count the aggregated number of samples in each grid using start point records to be a vector denoted as $b \in \mathbb{R}^m$. Then, we want to find some weight w which satisfies $Aw = b$. The constraint gives the relationship between the trajectory dataset and the OD dataset.

The framework of the constrained trajectory generator is shown in figure 5.4. The inputs of this method are query trajectories, and the outputs of this method are query trajectories and population density calculated using the OD dataset. The query trajectories and population density of destination will be fed into the model for training while the population density of origin is used to build the constraints.

The query trajectories will be compressed as a latent vector and a weight by the encoder of VAE. The latent vector is used to form the posterior distribution, while the weight is used to form a weighted Gaussian mixture. We sampling points from this posterior distribution to reconstruct the query trajectories using the decoder of VAE. At the same time, we are also sampling points from the weighed Gaussian mixture to reconstruct a larger trajectory dataset using the same decoder.

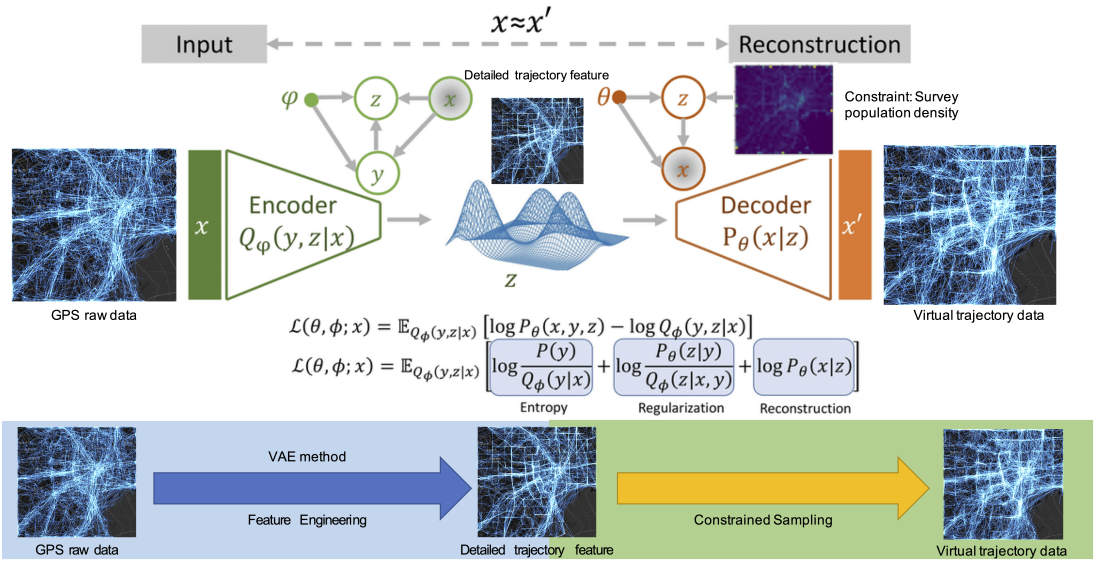


FIGURE 5.4: The framework of Constrained generator.

If we want to solve the issue that the number of reconstructed trajectories is the same with the population density, we need make sure every output weight w of encoder in VAE satisfy the constraint $Aw = b$. in this research, the constraint is built in a linear equality way, so we use a projection method to solve this constraint in a simple yet efficient way. For each output weight w of encoder in VAE, we could find the closest $w^\star$ which satisfy the constraint $Aw^\star = b$ by:

$$w^\star = (I - A^T(AA^T)^{-1}A)w + A^T(AA^T)^{-1}b \tag{5.5}$$

Where I is the identity matrix. We can prove Eq. 5.5 always hold when $rank(A) = m < n$.

Then the model is trained using a loss function contains three parts $L = L_{recon} + \alpha * L_{KL} + \beta * L_{density}$. The first part is the reconstruction error when this model tries to reconstruct the query trajectory dataset. The second part is the KL-divergence which penalize the learned latent vector to a simple unit Gaussian mixture. The third

part is the population density error calculated by reconstructed large trajectory dataset compared with the ground truth population density calculated by OD dataset. α, β are hyperparameters which adjust how much contribution of each part of loss we want to feed back into the model for a proper training.

The adaptive learning rate is often used to balance the convergence speed and the best convergence effect when training deep learning models. One of the most fundamental concepts of deep learning models is the backpropagation algorithm. In the training of the deep learning model, we usually need to first calculate the gradient of the loss function to each parameter in the deep learning model and then assign the loss to each parameter in the model according to the estimated gradient to achieve parameter update. When updating parameters, we also need to define a learning rate to adjust the amplitude of parameter updates, as shown in the figure below.

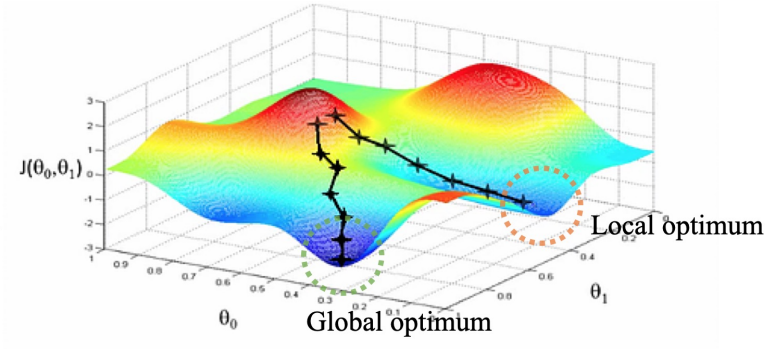


FIGURE 5.5: Illustration of gradient descend when training deep learning models.

The figure shows the visualization of gradient descends. The backpropagation algorithm can ensure that the model parameters move to a minor gradient direction with a high probability after each update. If the problem learned by deep learning is a simple single extreme value problem, then theoretically, the parameters of deep learning can always converge to that single extreme value. However, if the problem learned by the deep learning model is a relatively complex multi-extreme problem, then the model may not always converge to the global optimum in the end. In the figure, we have given a local optimum (orange dash circle) and a global optimum (green dash circle) and also given the two possible convergence directions of the deep learning models (black line). People have thought of many ways to make the deep learning model converge to the global optimum with a high probability.

There are many advantages to using an adaptive learning rate to adjust the learning rate during model training. Usually, we use a larger learning rate in the early stage of deep learning model training to quickly reach an optimal value. Nevertheless, since this best point is not always the global best, we usually still let the model jump around this best point with a larger learning rate. When the current optimal point is not the global optimal, a larger learning rate helps make the model jump out of the local optimal and move toward a smaller gradient. However, when we use a larger learning rate for a while, the training model still does not reach a direction with a smaller gradient. We start to reduce the learning rate because a lower learning rate helps us make fine-tuning nearby to speed up convergence. At the same time, the adaptive learning rate also helps prevent underfitting because we always give the model more opportunities to try different gradient directions.

5.3 Experiments

5.3.1 Data description

The primary data we used is navigation GPS data of human mobility of entire Japan. This dataset consists of extensive individual navigation GPS trajectories that contain information such as a Trip ID, timestamp, longitude, and latitude. We define the experimental area in this chapter as a rectangle of longitude from 139.5 to 139.9, latitude from 35.5 to 35.8, respectively, which can be seen as the center area of Tokyo. The following figure is a simple visualization to show the experimental area with human mobility GPS trajectories. Then this dataset is also used to simulate a heterogeneous OD dataset, which is shown in the figure 5.1.

5.3.2 Metrics

The metrics used for assessment are the most commonly used Mean Square Error (MSE), Earth Mover's Distance (EMD), and CityEMD. MSE is a most commonly used metrics for evaluate the Euclidean distance between two distributions.

$$MSE = \frac{1}{N}\sum_{1}^{N}(\hat{Y}^2 - Y)^2 \tag{5.6}$$

FIGURE 5.6: Visualization of the experimental area with human mobility GPS trajectory.

EMD is also known as the Wasserstein metric, which measures the distance of two probability distributions. Informally, we can consider the EMD as the minimum cost of turning one pile into the other.

$$
\begin{aligned}
d_w(\pi, \sigma) &= \sum_{r=1}^{n} d'_w(\pi(r), \sigma(r)) \\
d'_w(u, v) &= \begin{cases} \sum_{t=u}^{v-1} w_t & \text{if } u < v \\ d'_w(u, v) & \text{if } u > v \\ 0 & \text{Otherwise} \end{cases}
\end{aligned} \tag{5.7}
$$

CityEMD is proposed to measure the distance between two citywide human mobility pattern [27].

$$
CityEMD(t) = \sum_{o} dist(p(d|t + \Delta t, U_t^o), \hat{p}(\hat{d}|t + \Delta t, U_t^o)) \tag{5.8}
$$

5.3.3 Results

Training log is given in Figure 5.7, 5.8, 5.9.

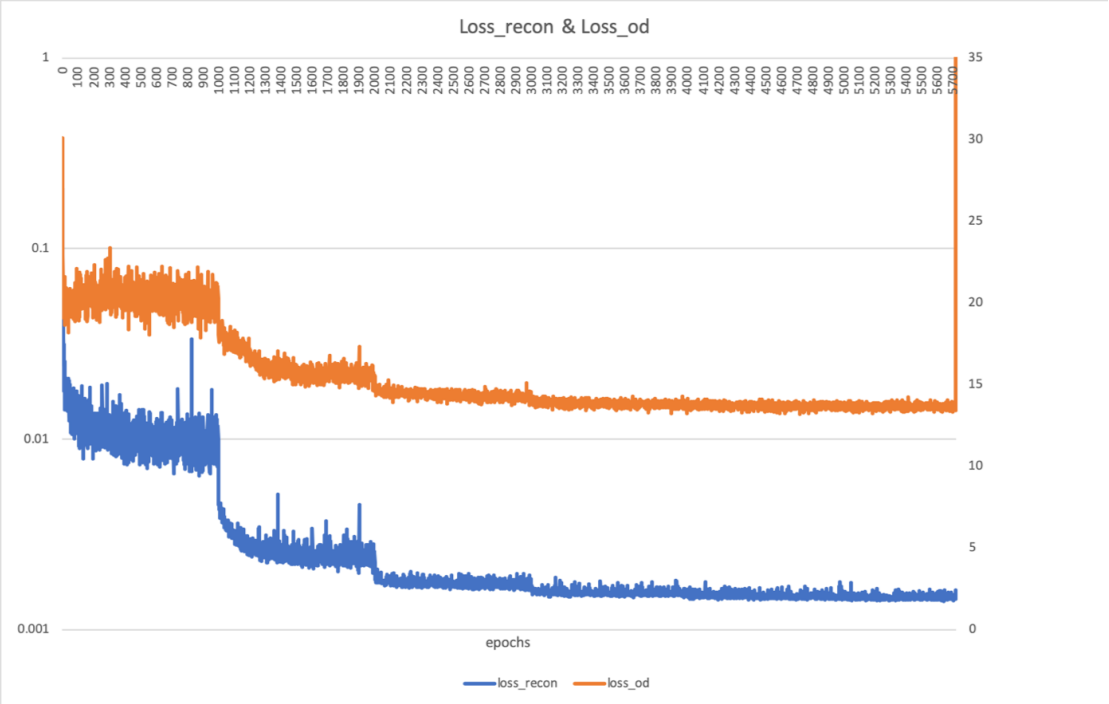


FIGURE 5.7: Reconstruction loss and OD distribution loss when training.

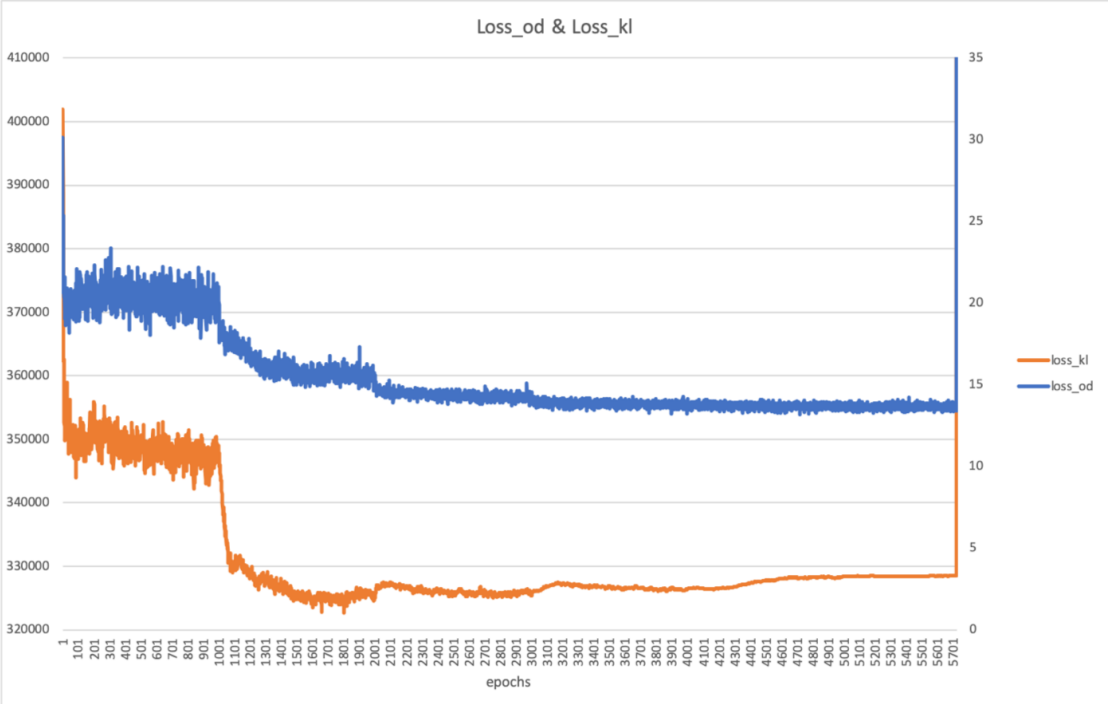


FIGURE 5.8: KL divergence loss and OD distribution loss when training.

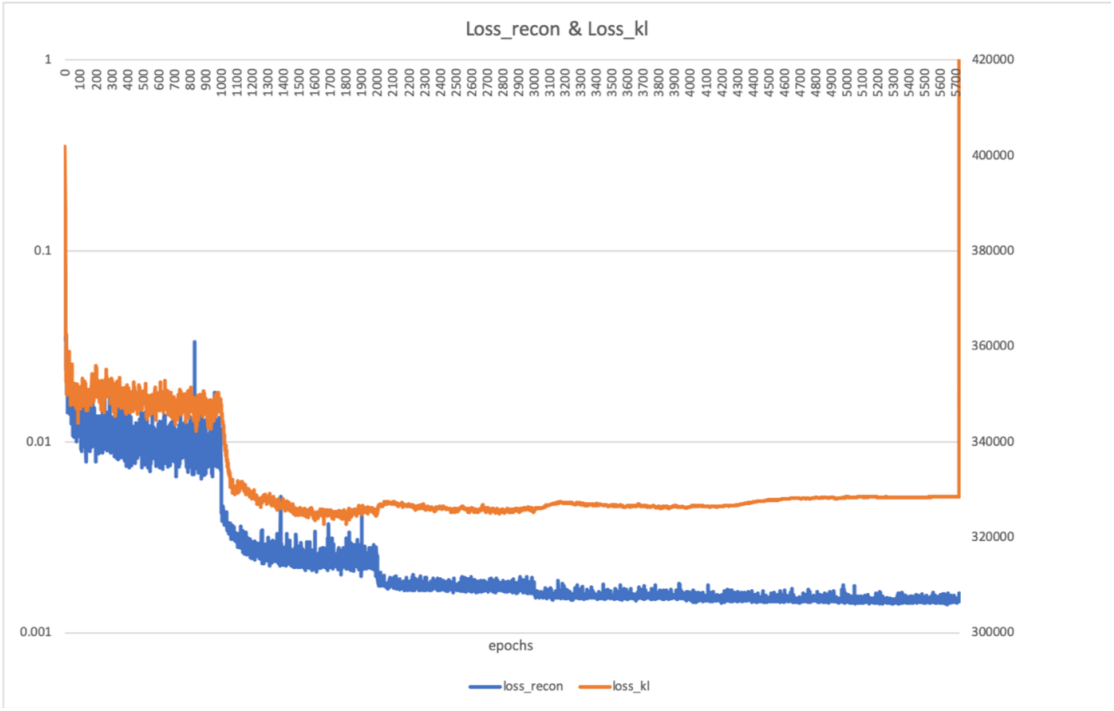


FIGURE 5.9: Reconstruction loss and KL divergence loss when training.

From training log from above figures, we know that 1) Better reconstruction accuracy leads to better bias reduction; 2) Better diversity leads to better bias reduction; 3) Better reconstruction accuracy leads to worse diversity.

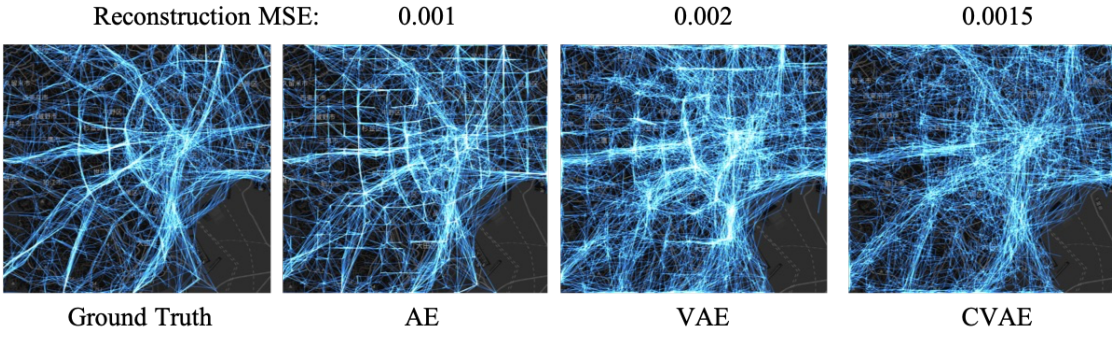


FIGURE 5.10: Visualization of ground truth and reconstruction results of AE, VAE, and CVAE.

TABLE 5.1: Effectiveness of a constrained model.

	Bias	Constrained VAE	Non-constrained VAE
MSE	11.993	13.6187	16.8160
EMD-1D	4.6312	3.4965	5.0104
EMD-2D	5.8085E4	6.1821E4	6.5668E4

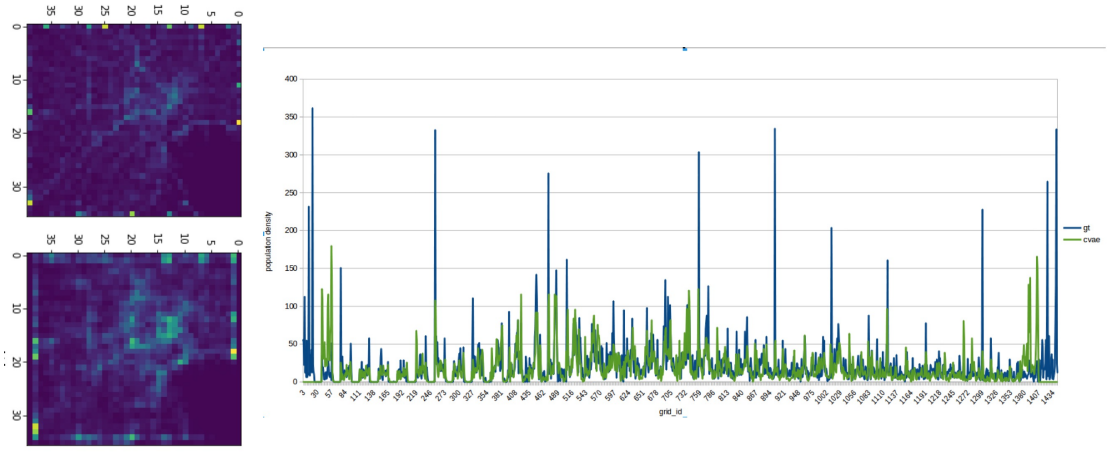


FIGURE 5.11: Visualization of population density of ground truth data (upper left), estimated data (bottom left), and comparison fo EMD.

We give some experimental results to show the effectiveness of constrained trajectory generator. From figure 8,9, we can see the comparison of AE, VAE, and CVAE use the metrics of MSE and CityEMD. The reconstruction MSE is used for measuring the ability of the model for reconstructing urban trajectories from points sampled in latent vector space.

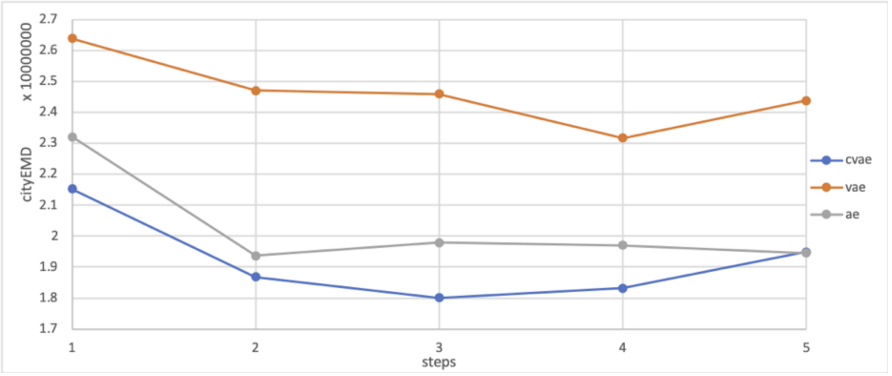


FIGURE 5.12: CityEMD comparison between results of AE, VAE, and CVAE.

We see that all models give acceptable accuracy when generating new trajectories. The bottom figure is the CityEMD of these three methods. CVAE gives the best performance which means that the global pattern of reconstructed large trajectory dataset is most similar with the ground truth data. In table 2, we show the effectiveness of a constrained model comparing with the non-constrained model. The bias is defined as the error of population density when evenly scaling up query trajectories. The constrained trajectory generator always performs better than a non-constrained model in all metrics. We also notice that the constrained trajectory generator gives worth performance than bias in

MSE and EMD-2D. This is caused by the reconstruction error introduced when we generating trajectories from latent vector space while evenly scaling up query trajectory dataset doesn't introduce any reconstruction error.

5.3.4 More evaluation

Evalution of different scenarios including: The number of query trajectories from 500 to maximum number (30,000) of trajectory dataset (from left to right direction). The number of target trajectories from 2,000 to maximum number (30,000) of trajectory dataset (from up to down direction). Different sampling rate from 3 hours to 6 hours. Every row in each figures gives a set of experiment: for example, in figure 5.13, (1) is the visualization of OD distribution of 2,000 ground truth trajectories. Meanwhile, (1.a), (1.b), and (1.c) shows the OD distribution of 2,000 generated virtual trajectories using 500, 1,000, and 2,000 trajectories as query trajectories respectively.

Comparison with metrics including: Reconstruction error (Recon), Relative distribution error (Relative dist), Absolute distribution error (Absolute dist), Reconstructed distribution error (Recon dist), Relative distribution error without boundary (Relative dist (no boundary)).

5.4 Conclusion

Constrained trajectory generator is shown to be effective comparing with previous methods. The current problem in this part is that we only conduct experiments with only some linear equality constraints, but there are many more complex and difficult constraints if we want utilize more heterogenous data sources to enrich the information learning by the model. The next step for this method is a generalized projection module for constrained trajectory generator.

TABLE 5.2: Sampling input every 6 hours

No.	Qeury	Input	Recon	Relative dist	Absolute dist	Recon dist	Relative dist (no boundary)
1	2000	500	1.6925	1.1816	2.3632	0.1116	1.0084
2	2000	1000	1.6509	1.0713	2.1426	0.1181	0.2966
3	2000	2000	1.6527	0.5335	1.067	0.1227	0.2208
4	5000	500	1.7767	0.5808	2.904	0.1124	0.2672
5	5000	1000	1.6554	0.5866	2.933	0.1196	0.2406
6	5000	2000	1.6598	0.6244	3.122	0.1141	0.2064
7	5000	5000	1.6645	0.5746	2.873	0.1593	0.1880
8	10000	500	1.5205	1.1063	11.063	0.1133	0.4315
9	10000	1000	1.6029	1.1942	11.942	0.1107	0.3422
10	10000	2000	1.6374	1.0834	10.834	0.1020	0.2775
11	10000	5000	1.6365	1.1551	11.551	0.1453	0.2325
12	30000	500	1.6351	3.3185	99.555	0.1082	1.1118
13	30000	1000	1.7032	2.9992	89.976	0.1094	0.8032
14	30000	2000	1.6485	2.9967	89.901	0.1180	0.5457
15	30000	5000	1.6907	3.1043	93.129	0.1661	0.4326
16	30218	5154	1.6710	3.1858	96.2685044	0.1559	0.4735

TABLE 5.3: Sampling input every 3 hours

No.	Qeury	Input	Recon	Relative dist	Absolute dist	Recon dist	Relative dist (no boundary)
17	2000	500	1.6768	1.4084	2.8168	0.1020	1.2450
18	2000	1000	1.6969	0.7081	1.4162	0.0979	0.5380
19	2000	2000	1.6530	0.5852	1.1704	0.1323	0.2892
20	5000	500	1.7042	0.6633	3.3165	0.1192	0.2838
21	5000	1000	1.7142	0.6006	3.003	0.1203	0.2165
22	5000	2000	1.6534	0.6013	3.0065	0.1285	0.2000
23	5000	5000	1.6901	0.6148	3.074	0.1637	0.1681
25	10000	500	1.5944	1.1095	11.095	0.1000	0.4639
26	10000	1000	1.6477	1.1038	11.038	0.1069	0.3505
27	10000	2000	1.7942	1.0908	10.908	0.1158	0.2447
28	10000	5000	1.6408	1.0832	10.832	0.1544	0.2573
29	10000	10000	1.7332	1.1432	11.432	0.2056	0.2282
30	30000	500	2.1805	3.3304	99.912	0.1100	1.2148
31	30000	1000	1.5872	2.8270	84.81	0.1212	0.7699
32	30000	2000	1.7558	2.9348	88.044	0.1112	0.5687
33	30000	5000	1.7161	3.0842	92.526	0.1658	0.4652
34	30000	10000	1.6852	3.1030	93.09	0.1981	0.3981
35	30218	10018	1.6467	3.0378	91.7962404	0.2139	0.4166

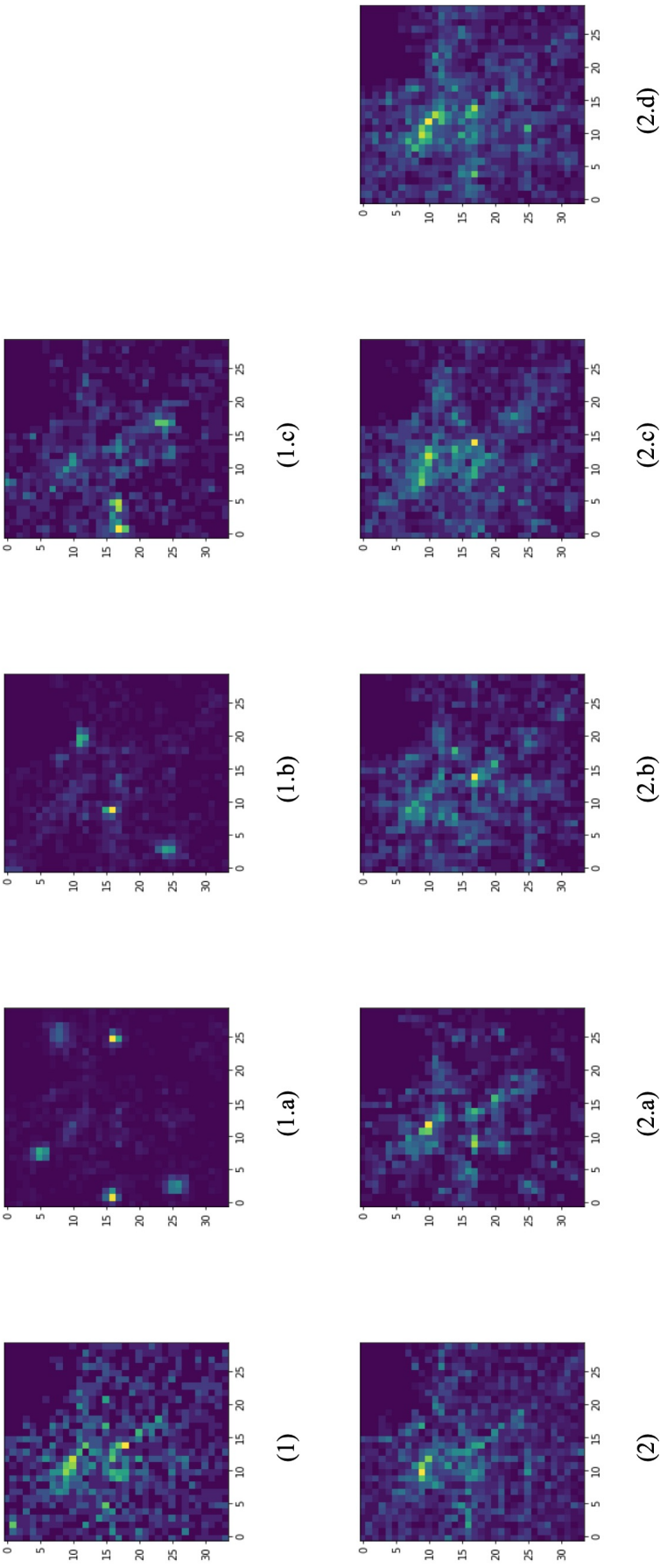


FIGURE 5.13: Visualization of OD distribution of reconstruction results when sampling trajectory every 6 hours (part 1).

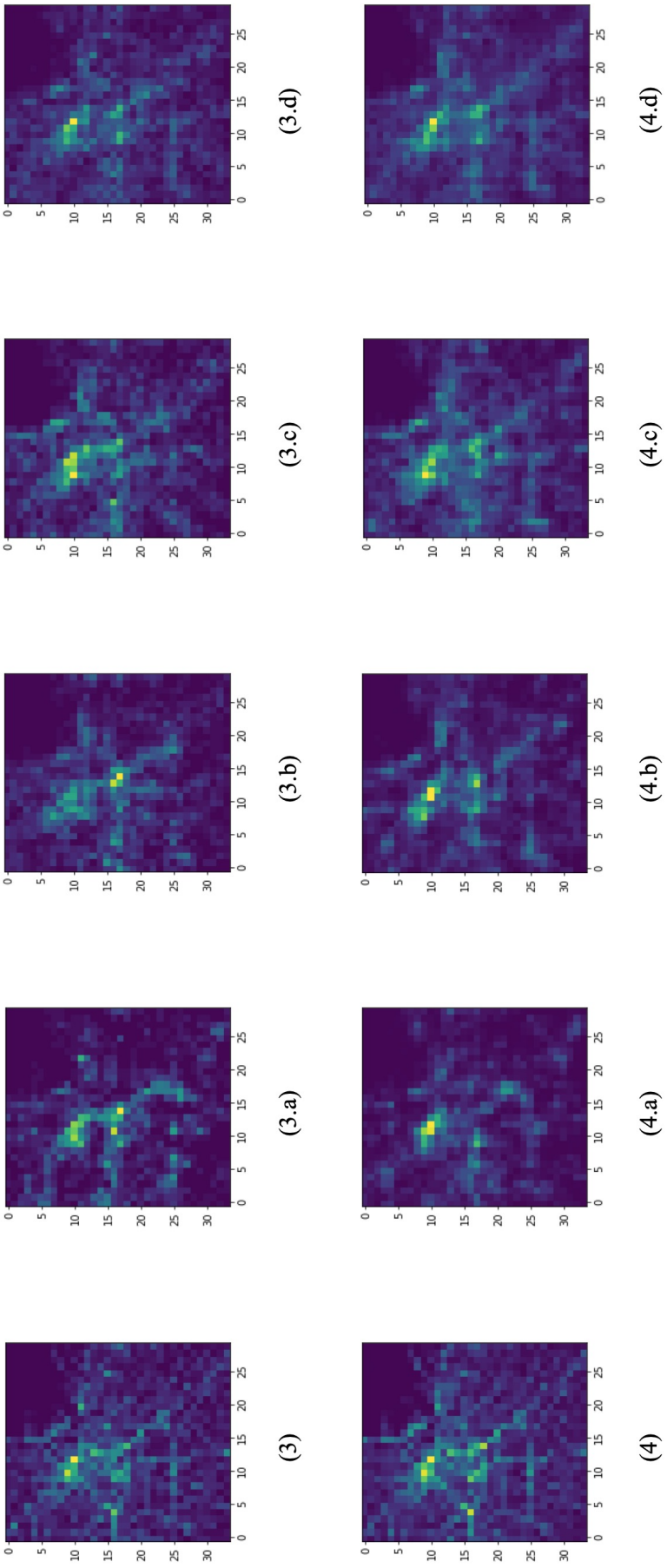


FIGURE 5.14: Visualization of OD distribution of reconstruction results when sampling trajectory every 6 hours (part 2).

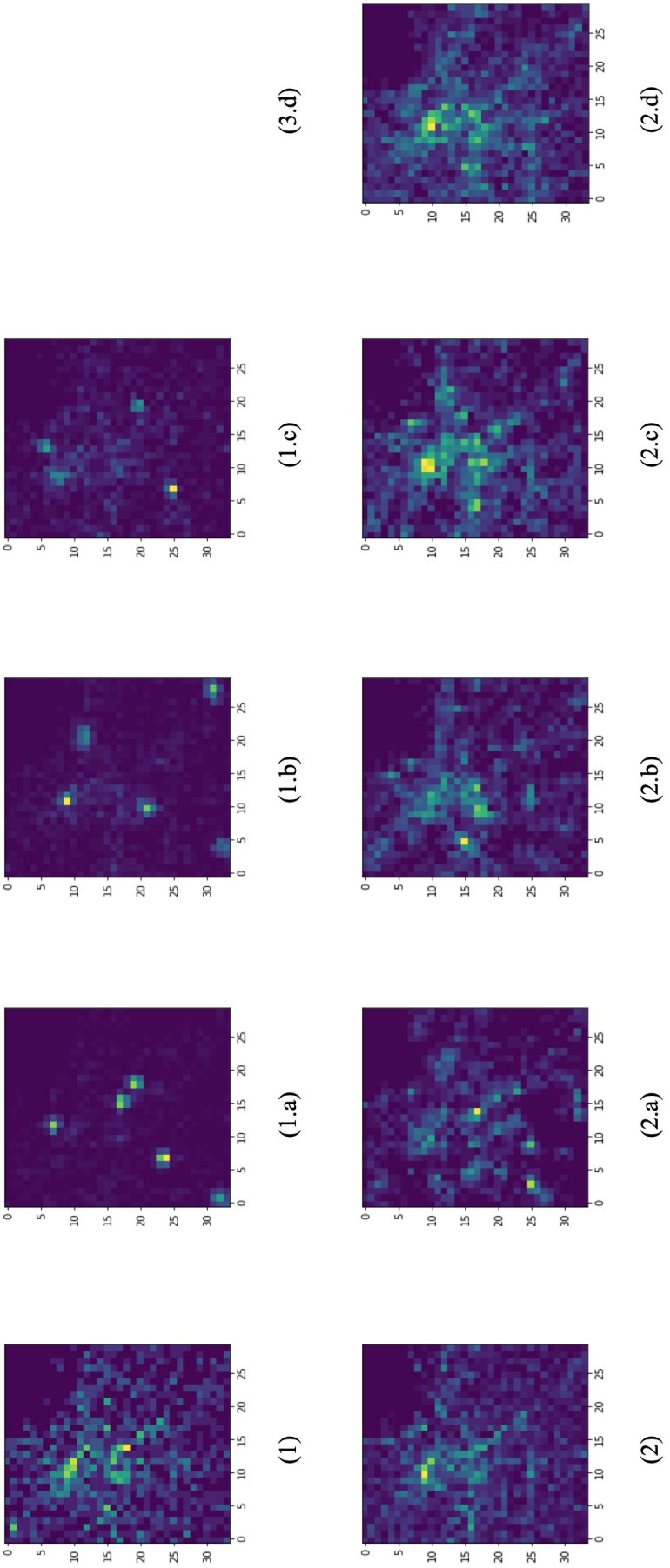


FIGURE 5.15: Visualization of OD distribution of reconstruction results when sampling trajectory every 3 hours (part 1).

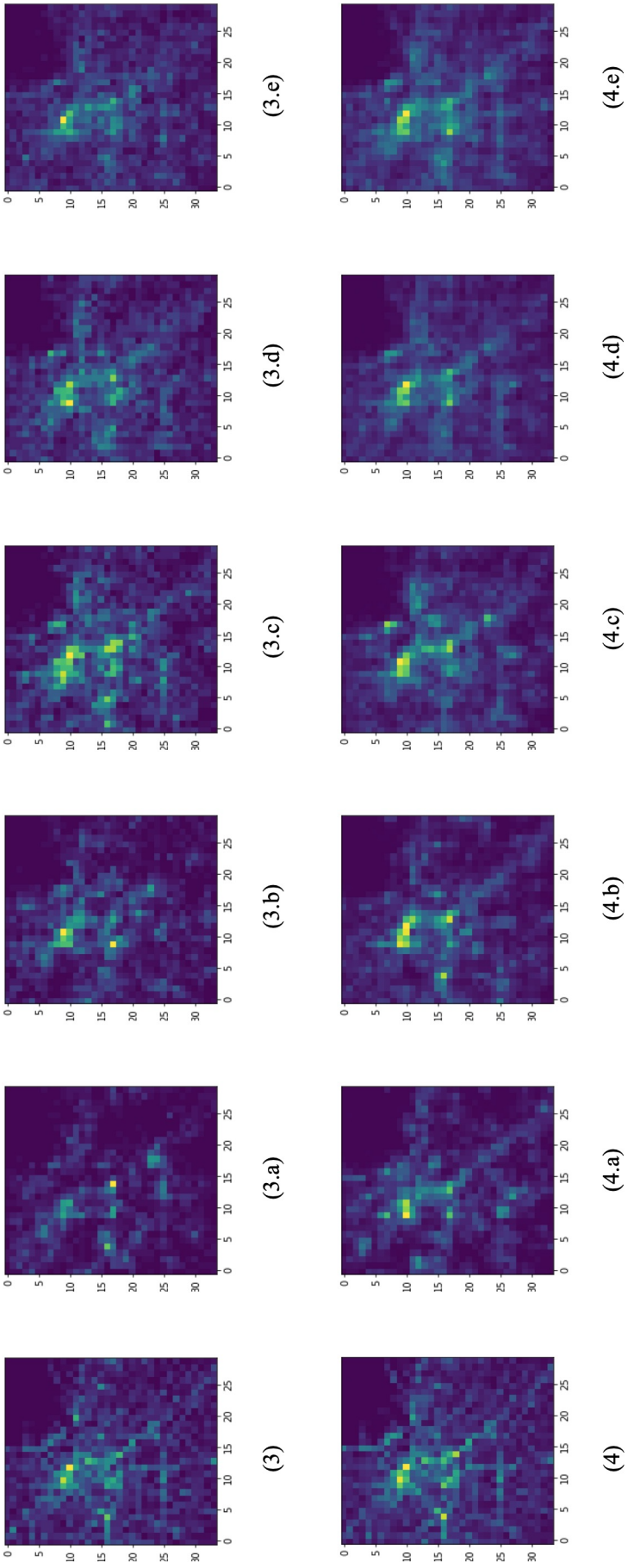


FIGURE 5.16: Visualization of OD distribution of reconstruction results when sampling trajectory every 3 hours (part 2).

Chapter 6

Conclusion and Future work

6.1 Concluding Remarks

We discussed and analyzed the two main issues: privacy protection and reduction of sampling bias and noise in the existing human mobility data existing in the city. At the same time, we also try to solve the above two problems from different angles and fully use the advantages of deep learning.

In the beginning, we tried to use Bayesian inference to understand the problem of human mobility estimation. We assume that any human mobility trajectory can find the corresponding hidden space distribution to represent the critical time-series information contained in the trajectory data. Under this assumption, as long as we know the joint probability distribution of the observed trajectory data and the hidden layer spatial distribution, we can estimate the human mobility within the city without directly using the observed trajectory data. We use LSTM to process the conversion between observation trajectory data and hidden space distribution in practical applications. Then, we use a variational generative model to generate more virtual trajectory data. The experimental results show that the generative model used has indeed achieved our goal. The virtual trajectory data sampled from the hidden space distribution has a certain degree of reconstruction accuracy and some diversity different from the observed trajectory. However, one limitation of this method is that no geographic information input is given. Hence, the newly generated virtual trajectory data also lack geographic information constraints. The trajectory points are sometimes located outside the road network.

In order to solve the limitations of the generative model mentioned above, the most intuitive way is to conduct a map matching to the newly generated trajectory data. However, we believe that using map matching as post-processing will change the original pattern of the virtual trajectory data. In order to verify our ideas, we did a simple experiment and used various trajectory similarity indicators to quantify the differences in patterns after post-processing. The experimental results show that the trajectory after this post-processing has changed by more than 20% compared with the original trajectory. Therefore, we want to avoid this simple use of map matching as a post-processing method to solve the above problems. At the same time, we also realize that it is difficult for us to find suitable indicators to quantify the authenticity of the newly generated virtual trajectory data. Because our observation trajectory data itself has sampling bias, even if we newly generate a different trajectory from the observation trajectory, we cannot directly think that this newly generated trajectory is unreal. In order to solve the above two problems, we refer to the work of traditional trajectory similarity and propose a retrieval-based method to generate new trajectories. We still use the deep learning model to encode the observation trajectory data to get a hidden space distribution. Then we use the same encoder to encode the entire trajectory database and then use the k-d tree to quickly search for historical trajectories similar to the observed trajectory to generate new virtual trajectory data. Under a similar framework, we compared the deep learning model with some traditional trajectory similarity methods. The experimental results show that the deep learning encoder is more capable of mining the time series characteristics of the trajectory itself and can distinguish different trajectories more efficiently. However, this method still has some limitations. For example, under this framework, we cannot know the weight of each observation trajectory in the entire citywide human mobility, so it is difficult for us to make better use of the limited observation trajectory data to reduce sampling bias.

Then we did not immediately propose a new framework to improve the above limitations but first introduced a differentiable projection method to construct a constrained deep learning model. Deep learning itself is a pure data-driven algorithm. Its advantage is that we can use a general algorithm framework to solve various problems without additional expert knowledge. However, in actual application scenarios, we sometimes know some expert knowledge in advance, and we want the deep learning model to learn within the constraints of this expert knowledge. That is, we want to implement a

constrained deep learning model. Inspired by some research in the field of constrained optimization, we found a differentiable projection method. Although this method cannot directly solve the linear constraint optimization problem, the point after the projection is closer to the entire feasible solution domain than the initial point. Then the constrained deep learning model can make full use of some expert knowledge without data-driven learning. Moreover, we use synthetic data to verify the proposed model, which provides a theoretical basis and application method for constructing a constrained human mobility estimation model later.

Finally, we give a straightforward application of the constrained deep learning in human mobility estimation. In this application, our purpose is to solve the problem that the retrieval-based model cannot directly give the weight of each observation trajectory. We believe it is necessary to introduce new data to solve this problem. Due to data limitations in actual implementation, we used simulated OD data, which is heterogeneous with human mobility trajectory data. However, it provides some additional information not in the trajectory data, such as population density. We carefully convert the trajectory data and establish a constraint relationship with the simulated OD data, and then apply the constrained deep learning proposed above for training. The experimental results show the effectiveness of constrained deep learning in human mobility estimation. It can maintain a sure accuracy of individual trajectory reconstruction and effectively reduce the sampling deviation of trajectory data by about 20% compared with the usual deep learning model. So far, we have completed the work from reconstructing individual trajectories to the estimation of citywide human mobility.

6.2 Future Work

We notice some limitations of methods proposed in current research, and we consider some improvements for some of them. However, there are still many possible directions for improvement. We have currently tried some frameworks for urban trajectory generators and show their performance compared with other methods. In the future, we will try to conduct more applications based on these proposed urban trajectory generators since we think each method has their strength and weakness. For example, a possible future work will build a latent semantic space of given urban trajectory data under the retrieval-based trajectory generator framework. We could use this latent semantic space

to retrieve trajectories that we need more specific and accurate. Besides, another future work will be building a more generalized constrained urban trajectory generator which subjects to some complex linear inequality constraints. It is worth building an integrated system of the current framework of human mobility estimation.

At the same time, we believe that data fusion is also a critical direction for human estimation. Because although we find that the current method has proposed a way to solve privacy infringement when using data, there is still a significant limitation for reducing the sampling bias and noise issue. The performance of our proposed methods to reduce the sampling bias of human mobility data is limited, owing to the use of a single data source. Therefore, we think there still has much room for improvement of this issue. In the future, we think a robust data fusion method is helpful to mine the information from different data sources to complete human mobility estimation with less sampling bias.

Appendix A

Appendix

Lemma 1. Given any $x \in \mathbb{R}^n, A \in \mathbb{R}^{m,n}$, and $rank(A) = m \leq n, b \in \mathbb{R}^m$. The closest point $x^\star$ subject to $Ax = b$ is

$$x^\star = (I - A^T(AA^T)^{-1}A)x + A^T(AA^T)^{-1}b$$

Proof. Suppose a subspace

$$S = \{x|Ax = b\} \tag{A.1}$$

For any $x \in \mathbb{R}^n$, we have

$$x = proj_S(x)Lproj_{S^\perp}(x) \tag{A.2}$$

Since S is the null space of A, $S^\perp$ is the orthogonal subspace of S:

$$\begin{aligned} S &= N(A) \\ S^\perp &= N(A)^\perp = C(A^T) \end{aligned} \tag{A.3}$$

Where $C(A^T)$ is the column space of the A^T. Suppose $\hat{x} \in \mathbb{R}^m$ is a vector in the column space of the A^T, so we have:

$$p = A^T\hat{x} \tag{A.4}$$

Given any $x \in \mathbb{R}^m$, a column vector $c = A^Tx$ should be projected into columns space of the A^T, which means that the error vector $e = c - p$ should be perpendicular with column space of the A^T:

$$Ae = 0 \tag{A.5}$$

So we have:

$$\begin{aligned} &A(c - A^T x) = 0 \\ &AA^T \hat{x} = Ac \\ &\hat{x} = (AA^T)^{-1} Ac \\ &p = A^T \hat{x} = A^T (AA^T)^{-1} Ac = proj_{S^\perp}(c) \end{aligned} \tag{A.6}$$

Therefore:

$$\begin{aligned} proj_{S^\perp} &= A^T (AA^T)^{-1} A \\ proj_S &= I - A^T (AA^T)^{-1} A \end{aligned} \tag{A.7}$$

Since $rank(A) = m$, we have $rank(AA^T) = m$, which is full rank, so AA^T is invertible. Since $x^\star$ is the projection of x onto the subspace S:

$$proj_S(x - x^\star) = 0 \tag{A.8}$$

We have:

$$(I - A^T (AA^T)^{-1} A)(x - x^\star) = 0 \tag{A.9}$$

Then:

$$x^\star = (I - A^T (AA^T)^{-1} A)x + A^T (AA^T)^{-1} b \tag{A.10}$$

■

Conbergence of projection method for linear inequality constraints

Lemma 4. For any $x \in \mathbb{R}^n$,

$$f(P_\alpha x) \leq f(x) - (\frac{2}{\alpha} - 1)||P_\alpha x - x||^2 \tag{A.11}$$

Proof. Since $P_i x$ is the closest point to x in C_i,

$$||P_i P_\alpha x - P_\alpha x||^2 \leq ||P_i x - P_\alpha x||^2 = ||P_i x - x||^2 + ||x - P_\alpha x||^2 - 2 < P_i x - x, P_\alpha x - x > \tag{A.12}$$

Therefore,

$$\begin{aligned} f(P_\alpha x) &= \sum_{j=1}^{k} \lambda_j ||P_j P_\alpha Y - P_\alpha Y||^2 \\ &\leq \sum_{j=1}^{k} \lambda_j ||P_j x - x||^2 + ||x - P_\alpha x||^2 - 2 < Px - x, P_\alpha x - x > \\ &= f(x) - (\frac{2}{\alpha} - 1)||P_\alpha x - x||^2 \end{aligned} \tag{A.13}$$

■

Define $g_\alpha : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ as $g_\alpha(x) = ||P_\alpha x - x||^2$. Let G be the set of minimizers of f. It is clear that f is a convex function since it is a positive combination of distances to closed convex sets. Theorem 1. $F = G$.

Proof. (1) $G \subset F$.

Take $x \in G$, so

$$f(x) - f(Px) \leq 0$$

From Lemma 4,

$$0 \leq g_1(x) \leq f(x) - f(Px)$$

(2) $F \subset G$.

Take $z \in F, x \in \mathbb{R}^n$.

Assume, by negation,

$$f(x) < f(z)$$

Consider the level set

$$A = \{y \in \mathbb{R}^n : f(y) \leq f(x)\}$$

A is closed and convex, because of the continuity and convexity of f.

Let y^0 be the closest point to z in A. By Lemma 4,

$$f(Py^0) \leq f(y^0)$$

The definition of y^0 implies now

$$||Py^0 - z|| \geq ||y^0 - z||$$

from Eq. 4.12,

$$||Py^0 - z|| = ||Py^0 - Pz|| \leq ||y^0 - z||$$

So

$$||Py^0 - z|| = ||y^0 - z||$$

Thus

$$Py^0 = y^0$$

Therefore

$$||y^0 - z|| = ||Py^0 - Pz|| \leq \sum_{i=1}^{r} \lambda_i ||P_i y^0 - P_i z|| \leq ||y^0 - z||$$

Then

$$||P_i y^0 - P_i z|| = ||y^0 - z||$$

for any i.

We can get

$$P_i z - z = P_i y^0 - y^0$$

So

$$f(z) = f(y^0) \leq f(x)$$

a contradiction.

So $f(z) \leq f(x)$ for any $x \in \mathbb{R}^n$, that is to say $z \in G$.

Let F_α be the set of fixed points of P_α.

From the definition of P_α that $F_\alpha = F$ for any $\alpha > 0$.

We define a feasible points by

$$C = \cap_{i=1}^{r} C_i \neq \Phi$$

We will show that $F = C$.

(1) Take $z \in F, x \in C$.

Since $x \in C, P_i x = x$ and $||P_i x - P_j x|| = 0$ for any i, j.

As in Lemma 3,

$$0 =< x - Pz, z - Pz >=< Px - Pz, z - Pz >\leq -\sum_{i=1}^{r}\sum_{j=1}^{r} \lambda_i \lambda_j ||P_i z - P_j z||^2$$

So

$$P_i z = P_j z$$

for all i, j

Then

$$z = \sum_{i=1}^{r} \lambda_i P_i z = P_i z$$

for any j, implying $z \in C$.

(2) Obviously, $C \subset F$. ■

Theorem 2. If $\{x^k\}$ defined by (11) is bounded, then it converges for any $x^0 \in \mathbb{R}^n$, and $F \neq \Phi$.

Proof. If $\{x^k\}$ is bounded there exists a convergent subsequence

$$x^{k_j} \to x, j \to \infty \tag{A.14}$$

So

$$g_\alpha(x^{k_j}) \to g_\alpha(x), j \to \infty \tag{A.15}$$

Using Lemma 4, we have

$$g_\alpha(x^{k_j}) \leq \frac{\alpha}{2-\alpha}[f(x^{k_j}) - f(x^{k_j+1})] \tag{A.16}$$

Since $f(x^k)$ is decreasing and bounded below by 0, $f(x^k) - f(x^{k+1})$ tends to 0.

So

$$g_\alpha(x) = 0 \Rightarrow P_\alpha x = x \Rightarrow x \in F$$

So

$$F \neq \Phi$$

Given a small real number $\epsilon > 0$, let

$$||x^{k_j} - x|| < \epsilon$$

For any $m > k_j$, we have

$$||x^m - x|| \leq ||x^{k_j} - x|| < \epsilon$$

So we get

$$x^k \to x \tag{A.17}$$

■

Bibliography

[1] Masayuki Morikawa. Population density and efficiency in energy consumption: An empirical analysis of service establishments. *Energy Economics*, 34(5):1617–1622, 2012.

[2] Zhaoyuan Yu, Linwang Yuan, Wen Luo, Linyao Feng, and Guonian Lv. Spatio-temporal constrained human trajectory generation from the pir motion detector sensor network data: a geometric algebra approach. *Sensors*, 16(1):43, 2016.

[3] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):1–55, 2014.

[4] Zipei Fan, Xuan Song, Ryosuke Shibasaki, Tao Li, and Hodaka Kaneda. Citycoupling: bridging intercity human mobility. In *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing*, pages 718–728, 2016.

[5] Andy Yuan Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *2013 IEEE 29th international conference on data engineering (ICDE)*, pages 254–265. IEEE, 2013.

[6] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

[7] Albert-Laszlo Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207–211, 2005.

[8] Dirk Brockmann, Lars Hufnagel, and Theo Geisel. The scaling laws of human travel. *Nature*, 439(7075):462–465, 2006.

[9] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *nature*, 453(7196):779–782, 2008.

[10] Andrea Hess, Karin Anna Hummel, Wilfried N Gansterer, and Günter Haring. Data-driven human mobility modeling: a survey and engineering guidance for mobile networking. *ACM Computing Surveys (CSUR)*, 48(3):1–39, 2015.

[11] Josh Broch, David A Maltz, David B Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, 1998.

[12] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seong Joon Kim, and Song Chong. On the levy-walk nature of human mobility. *IEEE/ACM transactions on networking*, 19(3):630–643, 2011.

[13] Kyunghan Lee, Seongik Hong, Seong Joon Kim, Injong Rhee, and Song Chong. Slaw: Self-similar least-action human walk. *IEEE/ACM Transactions On Networking*, 20(2):515–529, 2011.

[14] Joy Ghosh, Sumesh J Philip, and Chunming Qiao. Sociological orbit aware location approximation and routing (solar) in manet. *Ad Hoc Networks*, 5(2):189–209, 2007.

[15] Frédéric Morlot, Salah Eddine Elayoubi, and François Baccelli. An interaction-based mobility model for dynamic hot spot analysis. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[16] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 239–252, 2012.

[17] Francesco Calabrese, Giusy Di Lorenzo, and Carlo Ratti. Human mobility prediction based on individual and collective geographical preferences. In *13th international IEEE conference on intelligent transportation systems*, pages 312–317. IEEE, 2010.

[18] Amit Jardosh, Elizabeth M Belding-Royer, Kevin C Almeroth, and Subhash Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proceedings of*

the 9th annual international conference on Mobile computing and networking, pages 217–229, 2003.

[19] Fan Bai, Narayanan Sadagopan, and Ahmed Helmy. Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 2, pages 825–835. IEEE, 2003.

[20] Mirco Musolesi and Cecilia Mascolo. Designing mobility models based on social network theory. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(3):59–70, 2007.

[21] Vincent Borrel, Franck Legendre, Marcelo Dias De Amorim, and Serge Fdida. Simps: Using sociology for personal mobility. *IEEE/ACM transactions on networking*, 17(3):831–842, 2008.

[22] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[23] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and pois. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 186–194. ACM, 2012.

[24] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.

[25] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, Teerayut Horanont, Satoshi Ueyama, and Ryosuke Shibasaki. Modeling and probabilistic reasoning of population evacuation during large-scale disaster. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239. ACM, 2013.

[26] Renhe Jiang, Xuan Song, Zipei Fan, Tianqi Xia, Quanjun Chen, Qi Chen, and Ryosuke Shibasaki. Deep roi-based modeling for urban human mobility prediction.

Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2(1):14, 2018.

[27] Zipei Fan, Xuan Song, Ryosuke Shibasaki, and Ryutaro Adachi. Citymomentum: an online approach for crowd behavior prediction at a citywide level. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 559–569. ACM, 2015.

[28] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98. ACM, 2011.

[29] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, and Ryosuke Shibasaki. Intelligent system for urban emergency management during large-scale disaster. In *AAAI*, pages 458–464, 2014.

[30] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, and Ryosuke Shibasaki. Prediction of human emergency behavior and their mobility following large-scale disaster. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 5–14. ACM, 2014.

[31] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, Ryosuke Shibasaki, Nicholas Jing Yuan, and Xing Xie. A simulator of human emergency mobility following disasters: Knowledge transfer from big disaster data. In *AAAI*, pages 730–736, 2015.

[32] Quanjun Chen, Xuan Song, Harutoshi Yamada, and Ryosuke Shibasaki. Learning deep representation from big and heterogeneous data for traffic accident inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[33] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[34] Yaxiang Fan, Gongjian Wen, Deren Li, Shaohua Qiu, and Martin D Levine. Video anomaly detection and localization via gaussian mixture fully convolutional variational autoencoder. *arXiv preprint arXiv:1805.11223*, 2018.

[35] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and

captions. In *Advances in neural information processing systems*, pages 2352–2360, 2016.

[36] Jonas Mueller, David Gifford, and Tommi Jaakkola. Sequence to better sequence: continuous revision of combinatorial structures. In *International Conference on Machine Learning*, pages 2536–2544. PMLR, 2017.

[37] Shin Asakawa. Comparison between variational autoencoder and encoder-decoder models for short conversation. 2017.

[38] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

[39] M Ehsan Abbasnejad, Anthony Dick, and Anton van den Hengel. Infinite variational autoencoder for semi-supervised learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 781–790. IEEE, 2017.

[40] Mogeng Yin, Madeleine Sheehan, Sidney Feygin, Jean-François Paiement, and Alexei Pozdnoukhov. A generative model of urban activities from cellular data. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1682–1696, 2018.

[41] Mitra Baratchi, Nirvana Meratnia, Paul JM Havinga, Andrew K Skidmore, and Bert AKG Toxopeus. A hierarchical hidden semi-markov model for modeling mobility data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 401–412, 2014.

[42] Kun Ouyang, Reza Shokri, David S Rosenblum, and Wenzhuo Yang. A non-parametric generative model for human trajectories.

[43] S PERMISSION. Generative and discriminative classifiers: Naive bayes and logistic regression. 2005.

[44] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[45] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[46] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[47] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531. IEEE, 2011.

[48] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. *SLT*, 12(234-239):8, 2012.

[49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[50] Filippo Simini, Marta C González, Amos Maritan, and Albert-László Barabási. A universal model for mobility and migration patterns. *Nature*, 484(7392):96–100, 2012.

[51] Yuhao Yao, Haoran Zhang, Jinyu Chen, Wenjing Li, Mariko Shibasaki, Ryosuke Shibasaki, and Xuan Song. Mobsimilarity: Vector graph optimization for mobility tableau comparison. *arXiv preprint arXiv:2104.13139*, 2021.

[52] Ron J Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. Sequence-to-sequence models can directly translate foreign speech. *arXiv preprint arXiv:1703.08581*, 2017.

[53] Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. Table-to-text generation by structure-aware seq2seq learning. *arXiv preprint arXiv:1711.09724*, 2017.

[54] Sunyan Gu and Fei Lang. A chinese text corrector based on seq2seq model. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2017 International Conference on*, pages 322–325. IEEE, 2017.

[55] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[56] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[57] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 617–628. IEEE, 2018.

[58] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

[59] Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

[60] Haoran Sun, Xiangyi Chen, Qingjiang Shi, Mingyi Hong, Xiao Fu, and Nicholas D Sidiropoulos. Learning to optimize: Training deep neural networks for interference management. *IEEE Transactions on Signal Processing*, 66(20):5438–5453, 2018.

[61] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8614–8618. IEEE, 2013.

[62] Li Xu, Jimmy S Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. *Advances in neural information processing systems*, 27: 1790–1798, 2014.

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[64] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5, 2001.

[65] Davide Castelvecchi. Can we open the black box of ai? *Nature News*, 538(7623): 20, 2016.

[66] Youshen Xia. An extended projection neural network for constrained optimization. *Neural Computation*, 16(4):863–883, 2004.

[67] Youshen Xia, Henry Leung, and Jun Wang. A projection neural network and its application to constrained optimization problems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 49(4):447–458, 2002.

[68] Xing-Bao Gao, Li-Zhi Liao, and Liqun Qi. A novel neural network for variational inequalities with linear and nonlinear constraints. *IEEE transactions on neural networks*, 16(6):1305–1317, 2005.

[69] Fei Han, Qing-Hua Ling, and De-Shuang Huang. Modified constrained learning algorithms incorporating additional functional constraints into neural networks. *Information Sciences*, 178(3):907–919, 2008.

[70] Jianshu Chen and Li Deng. A primal-dual method for training recurrent neural networks constrained by the echo-state property. *arXiv preprint arXiv:1311.6091*, 2013.

[71] Belhassen Bayar and Matthew C Stamm. On the robustness of constrained convolutional neural networks to jpeg post-compression for image resampling detection. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2152–2156. IEEE, 2017.

[72] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1796–1804, 2015.

[73] Belhassen Bayar and Matthew C Stamm. Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. *IEEE Transactions on Information Forensics and Security*, 13(11):2691–2706, 2018.

[74] Alvaro R De Pierro and Alfredo N Iusem. A simultaneous projections method for linear inequalities. *Linear Algebra and its applications*, 64:243–253, 1985.

[75] MZ Nashed. Continuous and semicontinuous analogues of iterative methods of cimmino and kaczmarz with applications to the inverse radon transform. In *Mathematical Aspects of Computerized Tomography*, pages 160–178. Springer, 1981.

[76] Sarah Harris and David Harris. *Digital design and computer architecture: arm edition*. Morgan Kaufmann, 2015.

[77] Joydeep Dutta and CS Lalitha. Optimality conditions in convex optimization revisited. *Optimization Letters*, 7(2):221–229, 2013.

[78] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[79] Kun Ouyang, Reza Shokri, David S Rosenblum, and Wenzhuo Yang. A non-parametric generative model for human trajectories. In *IJCAI*, pages 3812–3817, 2018.

[80] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[81] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[82] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. Autoencoder for words. *Neurocomputing*, 139:84–96, 2014.

[83] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, pages 436–440, 2013.

[84] Zhuotun Zhu, Xinggang Wang, Song Bai, Cong Yao, and Xiang Bai. Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 204:41–50, 2016.

[85] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[86] Charles W Fox and Stephen J Roberts. A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2):85–95, 2012.

[87] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[88] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[89] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[90] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350, 2015.

[91] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor, Prof. Ryosuke Shibasaki, for his continuous support of my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. Then I would like to thank Prof. Xuan Song in our laboratory for his encouragement and insightful comments. Also, my sincere thanks go to the rest of my thesis committee: Prof. Kaoru Sezaki, Prof. Yoshihide Sekimoto, Prof. Takeshi Deguchi, and Prof. Takahiko Kusakabe. Besides, I would like to thank Dr. Haoran Zhang, Dr. Zipei Fan, and Dr. Renhe Jiang for their advice in the past three years. I thank my fellow labmates in our group: Dr. Zhilin Guo, Dr. Tianqi Xia, Xiaodan Shi, Yuhao Yao, and Xudong Li. Last but not least, I would like to thank my parents, Gangsheng Huang and Yadong Fu, for their continuous and unwavering support of my life.

Publications

[1] Huang, Dou, Xuan Song, Zipei Fan, Renhe Jiang, Ryosuke Shibasaki, Yu Zhang, Haizhong Wang, and Yugo Kato. "A Variational Autoencoder Based Generative Model of Urban Human Mobility." In 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 425-430. IEEE, 2019.

[2] Jiang, Renhe, Xuan Song, Dou Huang, Xiaoya Song, Tianqi Xia, Zekun Cai, Zhaonan Wang, Kyoung-Sook Kim, and Ryosuke Shibasaki. "Deepurbanevent: A system for predicting citywide crowd dynamics at big events." In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2114-2122. 2019.

[3] Xia, Tianqi, Xuan Song, Dou Huang, Satoshi Miyazawa, Zipei Fan, Renhe Jiang, and Ryosuke Shibasaki. "Outbound behavior analysis through social network data: A case study of Chinese people in Japan." In 2017 IEEE International Conference on Big Data (Big Data), pp. 2778-2786. IEEE, 2017.