Discovering human mobility from mobile data : probabilistic models and learning algorithms

Weizhu Qian

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

utbm
université de technologie
Belfort-Montbéliard

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

WEIZHU QIAN

Discovering Human Mobility from Mobile Data: Probabilistic Models and Learning Algorithms

Thèse présentée et soutenue à Belfort, le 7 décembre 2020

Composition du Jury :

SIDI-MOHAMMED SENOUCI	Professeur à Université de Bourgogne	Président
GERMAIN FORESTIER	Professeur à Université de Haute Alsace	Rapporteur
YACINE OUZROUT	Professeur à Université Lumière Lyon 2	Rapporteur
YE-QIONG SONG	Professeur à Université de Lorraine	Examinateur
FABRICE LAURI	MCF à Université de Technologie de Belfort Montbéliard, UBFC	Examinateur
FRANCK GECHTER	MCF-HDR à Université de Technologie de Belfort Montbéliar, UBFC	Directeur de thèse

N° | X | X | X |

école doctorale **sciences pour l'ingénieur et microtechniques**

Title: Discovering Human Mobility from Mobile Data: Probabilistic Models and Learning Algorithms

Keywords: Machine Learning, Deep Learning, Probabilistic Models, Variational Inference, Mobile Data

Abstract:

Smartphone usage data can be used to study human indoor and outdoor mobility. In our work, we investigate both aspects in proposing machine learning-based algorithms adapted to the different information sources that can be collected. In terms of outdoor mobility, we use the collected GPS coordinate data to discover the daily mobility patterns of the users. To this end, we propose an automatic clustering algorithm using the Dirichlet Process Gaussian Mixture Model (DPGMM) so as to cluster the daily GPS trajectories. This clustering method is based on estimating probability densities of the trajectories, which alleviate the problems caused by the data noise. By contrast, we utilize the collected WiFi fingerprint data to study indoor human mobility. In order to predict the indoor user location at the next time points, we devise a hybrid deep learning model, called the Convolutional Mixture Density Recurrent Neural Network (CMDRNN), which combines the advantages of different multiple deep neural networks. Moreover, as for accurate indoor location recognition, we presume that there exists a latent distribution governing the input and output at the same time. Based on this assumption, we develop a Variational Autoencoder (VAE)-based semi-supervised learning model. In the unsupervised learning procedure, we employ a VAE model to learn a latent distribution of the input, the WiFi fingerprint data. In the supervised learning procedure, we use a neural network to compute the target, the user coordinates. Furthermore, based on the same assumption used in the VAE-based semi-supervised learning model, we leverage the Information Bottleneck theory to devise a Variational Information Bottleneck (VIB)-based model. This is an end-to-end deep learning model which is easier to train and has better performance. Finally, we validate these proposed methods on several public real-world datasets providing the results that verify the efficiencies of our methods as compared to other existing methods generally used.

Titre : Discovering Human Mobility from Mobile Data: Probabilistic Models and Learning Algorithms

Mots-clés : Machine Learning, Deep Learning, Probabilistic Models, Variational Inference, Mobile Data

Résumé :

Les données d'utilisation des smartphones peuvent être utilisées pour étudier la mobilité humaine que ce soit en environnement extérieur ouvert ou à l'intérieur de bâtiments. Dans ce travail, nous étudions ces deux aspects de la mobilité humaine en proposant des algorithmes de machine learning adapté aux sources d'information disponibles dans chacun des contextes. Pour l'étude de la mobilité en environnement extérieur, nous utilisons les données de coordonnées GPS collectées pour découvrir les schémas de mobilité quotidiens des utilisateurs. Pour cela, nous proposons un algorithme de clustering automatique utilisant le Dirichlet process Gaussian Mixture Model (DPGMM) afin de regrouper les trajectoires GPS quotidiennes. Cette méthode de segmentation est basée sur l'estimation des densités de probabilité des trajectoires, ce qui atténue les problèmes causés par le bruit des données. Concernant l'étude de la mobilité humaine dans les bâtiments, nous utilisons les données d'empreintes digitales WiFi collectées par les smartphones. Afin de prédire la trajectoire d'un individu à l'intérieur d'un bâtiment, nous avons conçu un modèle hybride d'apprentissage profond, appelé le Convolutional Mixture Density Recurrent Neural Network (CMDRNN), qui combine les avantages de différents réseaux de neurones profonds multiples. De plus, en ce qui concerne la localisation précise en intérieur, nous supposons qu'il existe une distribution latente régissant l'entrée et la sortie en même temps. Sur la base de cette hypothèse, nous avons développé un modèle d'apprentissage semi-supervisé basé sur le Variational Autoencoder (VAE). Dans la procédure d'apprentissage non supervisé, nous utilisons un modèle VAE pour apprendre une distribution latente de l'entrée qui est composée de données d'empreintes digitales WiFi. Dans la procédure d'apprentissage supervisé, nous utilisons un réseau de neurones pour calculer la cible, coordonnées par l'utilisateur. De plus, sur la base de la même hypothèse utilisée dans le modèle d'apprentissage semi-supervisé basé sur le VAE, nous exploitons la théorie des goulots d'étranglement de l'information pour concevoir un modèle basé sur le Variational Information Bottleneck (VIB). Il s'agit d'un modèle d'apprentissage en profondeur de bout en bout plus facile à former et offrant de meilleures performances. Enfin, les méthodes proposées ont été validées sur plusieurs jeux de données publics acquis en situation réelle. Les résultats obtenus ont permis de vérifier l'efficacité de nos méthodes par rapport à l'existant.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France

ACKNOWLEDGEMENT

This thesis would not have been possible without the guidance, insight and encouragement of my PhD advisor, Dr. Franck Gechter. Therefore I would like first to thank Dr. Franck Gechter for his efforts during my pursuit of PhD. I would like to thank Dr. Fabrice Lauri for his suggestions to my work. Also, many thanks for other colleges, the secretary of the computer science department, the secretary of CIAD, the secretary of PhD student office and other staffs working at CIAD and UTBM. I also would thank for the reviewers who are willing to spend their time to provide their valuable suggestions about my thesis.

I would like to thank Dr. Bowei Chen at University of Glasgow. It is a great time for me to work and discus scientific problems with him. I would like thank my land lord, Mr. Marc Stori. He had always been very kind and helpful to me. I also would like to thank the PhD students working together at UTBM, Chen Liu, Jian Zhang, Yong Shi, Tao Yang, Rongrong Liu, Hailong Wu, Hanqing Wang, Yang Zhou and many other students. We had a great time together in France, which prevent me being consumed by work completely. Some of them already graduated and I wish them all have a bright future. Thank other friends I met in France for their help for my work and life during the last three years.

Finally, I would like to thank my friends in China and my family for their long-time support and encouragement. Especially, I want to thank my mother for all her efforts to raise and educate me.

Contents

1

INTRODUCTION

1.1/ CONTEXT

Discovering human mobility using user data collected from smartphones has become a critical challenge especially in the recent similar context. Thanks to the recent advances in hardware and software technology, smartphone devices now integrated with various types of built-in sensors, such as cameras, accelerators, gyroscopes, Bluetooth, GPS modules and WiFi modules, etc., can offer various functions to users. Smartphone handsets are portable so that they can be used by their owners almost anytime and anywhere. For many people, smartphones have become important tools in their daily life. Consequently, the usage of smartphones reflects the daily life of the smartphone users as well. Therefore, some researchers attempt to take advantage of mobile data to study human behavior. **Mobile data** in this thesis is referring to smartphone usage data, including making phone calls, texting, taking photos, listening to music, etc.

In recent years, Location-Based Service (LBS) [Schiller et al., 2004] has been an important part of many applications, such as advertisement, social network and navigation. LBS is a type of technology that uses geographic information to provide users services. These services include health care, advertising, entertainment and indoor localization. Studying human mobility is essential for developing LBS applications. In this thesis, we want to study human mobility in both outdoor and indoor environment with mobile data. Hence, among all the aforementioned usage data, the location-related data attracts our attention the most. In literature, there are various types of data one can use for studying human mobility, such as GPS, WiFi and cell-IDs [Trevisani et al., 2004], [Bazzani et al., 2010], [Lin et al., 2005], [Zheng et al., 2008],

[Yavaş et al., 2005], [Su et al., 2000].

The Global Positioning System (GPS) is a satellite-based navigation system developed and owned by the United States, which can provide both geological and temporal information when users stay outdoors [Hofmann-Wellenhof et al., 2012]. The advantage of using GPS data is that it is more convenient than other indirect methods, such as cell-ID based methods and WiFi based methods which need further interpretation to acquire geological information. However, the drawback is that GPS modules do not work well in an indoor environment.

Fortunately, in recent decades, wireless networking technology has rapidly developed so we can consider WiFi-based localization techniques as an alternative. WiFi is a type of wireless networking technologies based on the IEEE 802.11 family of standards, which is used to local area networking and internet access [Gast, 2005]. When the WiFi module in a smartphone is turned on, it will automatically scan the WiFi access points (WAPs) near the device. Then, the WiFi scan list will show the Received Signal Strength Indicator (RSSI) values of each detected WAP. Normally, the RSSI values are lower as the WAP are farther from the device. Thank to this characteristic, we can localize the user position based on the corresponding RSSI values. The data obtained via such method is called **WiFi fingerprint** data.

To acquire GPS and WiFi fingerprint data, researchers can resort to the crowd-sourcing technique [Brabham, 2013]. The crowd-sourcing technique, in the context of this thesis, is to use the help of voluntary smartphone users to collect a large scale of data from a large group of users. This kind of databases can also be regarded as Big Data. Usually, more data means more information. Hence, training models with more data results in better results.

In order to build a large database more easily, a widely used approach in previous research work is to launch a campaign by asking volunteers to install the ad hoc applications developed by researchers on their smartphone devices. Such applications includes Mobile Data Challenge [Laurila et al., 2012], [Laurila et al., 2013], Device Analyzer [Wagner et al., 2013], [Wagner et al., 2014], UJIIndoorLoc [Torres-Sospedra et al., 2014] and Tampere [Lohan et al., 2017b]. These kinds of applications usually are designed to record the smartphone device usage. By contrast with traditional data collecting methods, crowd-sourcing does not need stand-alone

devices, i.g., GPS devices, to record human behavior. Instead, the practitioners can take advantage of the built-in sensors to collect user behavior data so that the normal daily life of the user will not be affected. As a result, the obtained data are more reliable for reflecting real human behavior.

Finally, with the access to the GPS and WiFi fingerprint data collected from smartphone users, the main objective of this thesis is to discover human mobility from the data. To this end, we will propose several machine learning and deep learning based methods in our work.

1.2/ Main Issues of the Thesis

In this thesis, the main goal is to discover the user mobility from the collected smartphone usage data. In order to have a comprehensive understanding of human mobility, we need to investigate both indoor and outdoor mobility of smartphone users. The data utilized in our work are GPS data and WiFi fingerprint data. When studying outdoor user mobility, using GPS data is more convenient though WiFi hotspots can be detected outdoors in some cases. As for studying indoor user mobility using WiFi fingerprint data is a feasible choice.

1.2.1/ Discovering Daily Mobility Patterns from GPS data

We want first to shed some light on the outdoor mobility of smartphone users. Because by doing this, we can have general knowledge of human mobility and behavior. In this work, in particular, we focus on discovering the daily mobility of the users. We believe that daily mobility can reflect life styles of smartphone users, which will help us to understand human behavior better. Nowadays, GPS modules are widely built in smartphone devices so as to provide the geographical location information for location-based services, such as navigation, advertisement and entertainment. Naturally, we can leverage these GPS modules to collect GPS data from smartphones in order to study human mobility.

In order to investigate daily activity patterns of people more thoroughly and precisely, we choose to resort to a considerable amount of GPS data enables us to study the human mobility at a large spatial and temporal scale. Thus for the GPS data-based method, we

want to analyze these patterns along a relatively long period (in our case, up to several months). Here daily mobility patterns refer to the most common trajectories users pass every day. Also, in order to make our method more convincing, we need to verify our method on sufficient user data. It means that we need to adopt a dataset collected from a number of different users. In our experiments, we take advantage of the Mobile Data Challenge (MDC) database [Laurila et al., 2012], [Laurila et al., 2013].

In this thesis, especially, we choose to study user mobility at the time slot of one day. It is because we argue that human mobility repeat daily, e.g., home → work place/school → home. In our work, we aim to discover mobility patterns in particular. Here, **mobility patterns** refer to the common trajectories used by users. Therefore, this can be regarded as a clustering daily trajectories problem. We can apply some machine learning techniques to this issue. A **trajectory** here is a set of GPS data points recording the mobility of the smartphone user during a certain time period. In this task, we do not treat these data points as sequences.

Moreover, we also should be aware of that the trajectories of users vary largely in space and time. For example, we may stay home on weekends and go to work on work days. Or on the way we go back to home, we may take a detour to go shopping. Consequently, these behaviors cause the uncertainty of human mobility. This issue will make some conventional clustering techniques, such as K-means, DBSCAN and Gaussian Mixture Models, unsuitable for this task.

By clustering user daily trajectories, we can understand human behavior, analyze people daily routines and activities, arrange better advertising strategies and analyze specific regions. In order to discover the daily mobility patterns, we propose a Dirichlet Process Gaussian Mixture Model-based model for clustering daily user GPS trajectories.

1.2.2/ WiFi Fingerprint-based Location Prediction

Apart from the GPS data-based outdoor mobility problem, we also want to address the indoor user mobility issues by using smartphone usage data. However, the difficulty is that GPS modules are malfunctional when users stay indoor, thus we cannot use GPS data to model indoor user mobility.

As previously mentioned, there are fortunately alternative methods. Nowadays, WiFi ac-

cess points are widely installed in modern buildings so as to provide Internet connection. And the signal strength of WiFi access points is related to the physical distance between the devices and the WiFi hotspots. Naturally, we can use this property for indoor localization.

In our work, we want to utilize WiFi fingerprint data for accurate location prediction. Here ”accurate” means that we will use the real user coordinates (which can be seen as the target of a regression task), instead of building IDs or floor IDs (which can be seen as the target of a classification task). More specifically, in our task, we want to do the **location prediction** task, which is to predict the next user location by using the WiFi fingerprint at the current time point. This task can be treated as a time series prediction. The input of this problem is the WiFi fingerprints and the target is the real coordinates of users. They are both sequential data. The WiFi fingerprint-based time series applications can be used for the services such as indoor navigation and advertisement.

Predicting user next location with WiFi fingerprints is a tricky problem because the input data is not easy to process. First, generally, for the purpose of providing high quality Internet connection, public buildings, such as office building, school buildings and super malls, are equipped with a large number of WiFi hotspots. Nevertheless, this also leads to the high dimensionality problem, which make the models prone to be overfitting and hard to converge. Furthermore, WiFi the signal fluctuations is detrimental to the accuracy of WiFi-based positioning methods. Theses challenges require us to adopt some feature detection techniques to extract the useful information from the input data.

Moreover, the relationship between RSSI values and actual user location is not trivial. Especially when the number of the WiFi access points and the amount of the data are large, using conventional machine learning methods is not easy to tune [Nowicki et al., 2017]. Thus a better approach is to utilize a model scalable enough, for example, a deep learning model. Therefore, in order to tackle with the aforementioned problems, we resort to the advanced deep learning techniques, to propose a novel deep learning-based method in our work, which is called the Convolutional Mixture Density Recurrent Neural Network.

1.2.3/ WiFi Fingerprint-based Location Recognition

In this work, we want to improve the WiFi fingerprint-based method even further, which means interpreting the WiFi fingerprints into real user coordinates more accurately. We treat this problem as a high dimensional regression task whose learning targets are numerical values. This task can be supervised, which means the targets are learned directly from the input, or semi-supervised, which means the targets are learned from a representation of the input. Accordingly, this problem is named as **location recognition** in this work.

To solve this problem, we can use some conventional machine learning models, such as k-nearest neighbours, Decision Trees and Random Forests, etc. These methods attempt to model the relationship between the input and the output directly. However, the modeling accuracies will be largely affected by the noise of the data. This issue remains the same even for some probabilistic models, such as Mixture Density Networks (MDNs), Gaussian Processes (GPs) and Bayesian Neural Networks (BNNs).

In order to address this issue, we find that both the input of the model, i.e., WiFi fingerprints, and the target of the model, i.e., the user coordinates, are related to the actual user location. Based on this idea, we can utilize a latent distribution to connect the input with the output instead of directly modeling the relationship between the input and the output. By doing so, we can obtain the useful information for learning the task from the original WiFi fingerprint data so as to circumvent the overfitting problem and improve the modeling accuracy.

We can use Generative Latent Variable Models (GLVMs) to implement our idea. In our approach, we use a encoder-decoder structure. We can either use a unsupervised deep learning model, for instance, a Variational Autoencoder (VAE), to do learn the latent representation first. Or we can use supervised deep learning model, for instance, a Variational Information Bottleneck (VIB), to proceed end-to-end learning.

Finally, we propose a Variational Autoencoder-based semi-supervised learning model for WiFi fingerprint-based accurate indoor positioning. Furthermore, we combine the Information Bottleneck method with Variational Inference to devise a new model, the Variational Information Bottleneck model for WiFi fingerprint-based accurate indoor location recognition.

1.3/ Main Contributions of the Thesis

To address the problems in last section, we propose several methods in this thesis. They are the Dirichlet Process Gaussian Mixture Model-based algorithm for clustering GPS trajectories, the Convolutional Mixture Density Recurrent Neural Network for sequential location prediction, the Variational Autoencoder-based semi-supervised model for location recognition and the Variational Information Bottleneck-based model for location recognition. The overview of our contributions in this thesis is exhibited in Fig. 1.1.

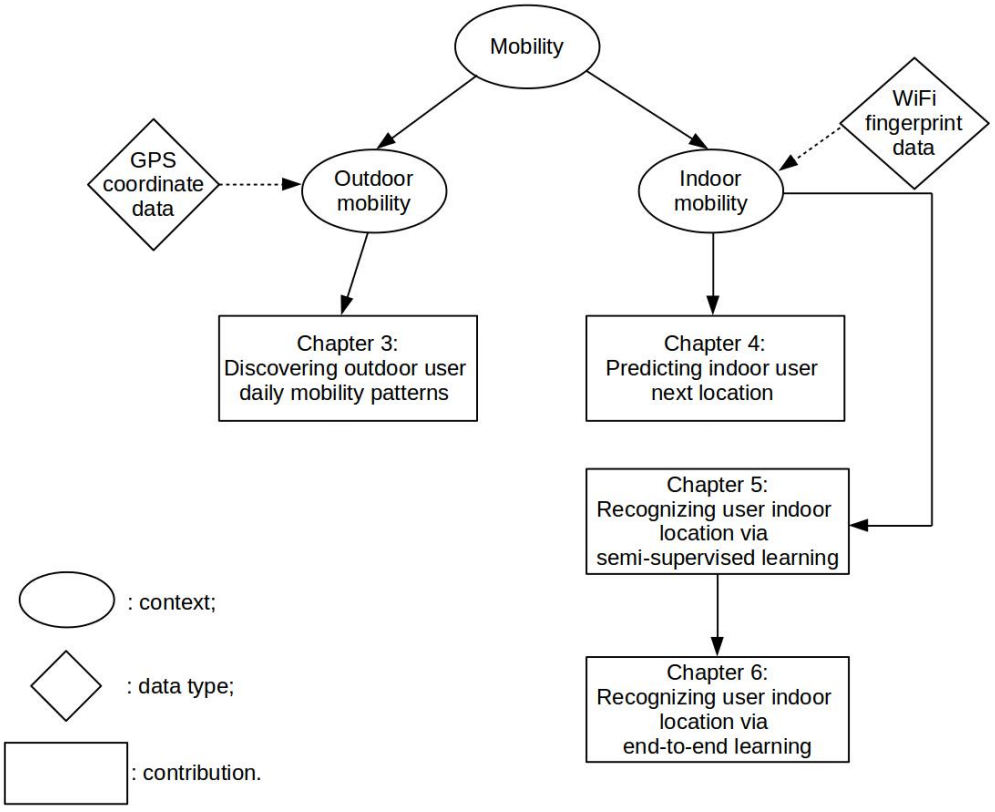


Figure 1.1: Overview of the thesis contributions.

1.3.1/ DPGMM-based Clustering Algorithm

In order to discover daily mobility patterns, we can cluster daily GPS trajectories. However,there are several issues when we try to cluster GPS data. One issue is that in some occasions the GPS modules do not function normally during the data collecting process,

for example, when the user is in a tunnel or stay indoor. Due to this, some part of the GPS data are missing, which cause the data sparsity problem. The other issue is that the GPS data are not distributed evenly space because the users stay in different places for different time periods. For instance, people stay at home or work places for longer time than in supermarkets.

For discovering daily mobility patterns, we propose a Dirichlet Process Gaussian Mixture Model (DPGMM)-based clustering method to cluster daily trajectories. This method has several advantages. First, this method adopts a probabilistic approach. It calculates the probability density of each trajectory and uses the Kullback-Leibler divergences as the clustering metrics instead of using the conventional Euclidean distance. By doing this, we can circumvent the data sparsity problem.

Furthermore, for estimating the probability densities more accurately, we use the Gaussian Mixture Model with a Dirichlet Process prior, this can avoid pre-defining the number of mixture models. Moreover, our algorithm is an automatic clustering algorithms, which means it does not need the prior knowledge of the cluster number.

1.3.2/ CMDRNN FOR SEQUENTIAL LOCATION PREDICTION

Since each element of the high dimensional features of the WiFi fingerprint data contributes equally (each element relates to one WiFi access point) to the user location information, Principal Component Analysis (PCA), a kind of dimension reduction method, is not suitable for such tasks [Nowicki et al., 2017]. Instead, we can resort to deep learning based techniques, for example, Autoencoders and Convolutional Neural Networks. In practice, we find that the Convolutional Neural Network outperforms the Autoencoder [Ibrahim et al., 2018].

Since the state transition information of the time-series data is implicit and the possible state space is very large, conventional approaches, such as Markov Models and Hidden Markov Models are not feasible for our tasks. Alternatively, we can use a deep learning model, such as a Recurrent Neural Network, to model the state transitions.

We also find that computing the user coordinates with a conventional neural network directly will result in severe overfitting. To solve this problem, we employ a Mixture Density Network in our model. The Mixture Density Network uses a set of mixed Gaussian dis-

tributions to sample the final output rather than compute it directly like a deterministic function. We deploy a Mixture Density Network at the final output layer of our model, which make the proposed model a Maximum Likelihood Estimation (MLE) model.

Finally, in order to predict user next location with WiFi fingerprints, through combining the advantages of the aforementioned deep learning models, we propose a novel deep learning-based model, called the Convolutional Mixture Density Recurrent Neural Network (CMDRNN). The proposed model is an end-to-end model, which means that it can be trained straightforwardly.

1.3.3/ VAE-BASED MODEL FOR LOCATION RECOGNITION

In order to localize user location with WiFi fingerprint data, we design a semi-supervised learning model based on Variational Autoencoders (VAEs) [Kingma et al., 2014b]. Our approach consists of two learning procedures, the first learning procedure is unsupervised learning which is used to learn a latent representation of the input data. In this procedure, we make use of a Variational Autoencoder to achieve the learning task.

The second learning procedure is a supervised learning process aiming to calculating the final user coordinates. To this end, we devise two neural network predictors. One predictor is a deterministic model whose loss function is root mean squared error and the other predictor is a stochastic model whose loss function is the negative log-likelihood.

1.3.4/ VIB-BASED MODEL FOR LOCATION RECOGNITION

We propose another deep learning model for accurate location recognition, which is called the Variational Information Bottleneck (VIB)-based model. This model combine the Information Bottleneck method and the Variational Inference. According to the Information Bottleneck theory, through learning a latent distribution, we can solely have the task-related information from the original data so as to alleviate the overfitting problem. However, implementing the Information Bottleneck method via neural network directly is not easy. Therefore, we leverage Variational Inference to derive a variational lower bound as the optimization target.

Similar to the VAE-based semi-supervised model, in the model we use a latent distribution

as the representation of the input as well. But the difference is the VIB-based model is supervised learning model. Its advantage is that it does not need an unsupervised learning process to learn the latent representation. It is an end-to-end model and able to learn directly the latent representation of the input data during the supervised learning process. This make its training process more straightforward than the VAE-based semi-supervised model.

1.4/ THESIS ORGANIZATION

The thesis includes 7 chapters and the remainder of the thesis is organized as follows:

Chapter 2: State of the Art. In this chapter, we review the previous work, both GPS-based methods and WiFi-based methods in literature.

Chapter 3: Discovering Daily Outdoor Mobility Patterns. In this chapter, we present the Dirichlet Process Gaussian Mixture Model-based algorithm for clustering daily GPS trajectory data to discover the outdoor mobility patterns of the users.

Chapter 4: Predicting Indoor Location with WiFi Fingerprints. In this chapter, we introduce the Convolutional Mixture Density Recurrent Neural Network for predicting indoor next location.

Chapter 5: Recognizing Indoor Location via Semi-Supervised Learning. In this chapter, we propose the Variational Autoencoder-based semi-supervised learning model for accurate user location recognition.

Chapter 6: Recognizing Indoor Location via End-to-End Learning. In this chapter, we develop the Variational Information Bottleneck-based model to compute the user location.

Chapter 7: Conclusions and Perspectives. In this chapter, we draw the final conclusions of the thesis and point out some possible research directions of the future work.

2

State of the Art

In this chapter, we will review prior works using GPS data and WiFi fingerprint data to study human mobility. Generally, the problems in theses works can be framed as different types of learning tasks, for instance, clustering, classification, regression and sequential prediction. Accordingly, we can adopt some techniques, such as conventional machine learning (ML) methods and deep learning (DL) methods, to address these problems. Machine learning [Bishop, 2006], [Murphy, 2012] is a class of algorithms that is used for data analysis, pattern recognition (PR), computer vision (CV), signal processing, natural language processing (NLP), etc. Thanks to the recent advances in hardware technology, computers are becoming more powerful and more adaptive to specific algorithms based on vectoral processing. Due to this reason, a group of machine learning algorithms, called deep learning [LeCun et al., 2015], [Goodfellow et al., 2016], has been rapidly developed. Deep learning models have very powerful modeling ability because they can have very deep structures with multiple hidden layers. In this chapter, we will introduce both machine learning and deep learning approaches previously applied to discover human mobility from mobile data.

2.1/ Discovering User Mobility Patterns from GPS Data

GPS data contains the information of latitudes and longitudes, which is able to directly provide relatively accurate the coordinates of users when users stay outdoors. Moreover, GPS data can record user mobility from a large range of space and time, which enables the researchers to unveil the human mobility patterns. Many previous researchers have developed various methods based on GPS data. In literature, previous research

such as [Lu et al., 2013], [Ye et al., 2012], [Lin et al., 2014], [Pirozmand et al., 2014], [Zheng, 2015] and [Cao et al., 2007], have studied human mobility by using GPS data collected form smartphone users. These works mainly focused on the tasks such as extracting significant visiting places, clustering trajectories, discovering travel sequences and so on.

2.1.1/ Discovering Frequently Visited Places

Through discovering frequently visited places, we can reveal how people behave in their daily life. To some extent, it can be seen as a task of clustering GPS data in our context. Clustering is a type of unsupervised learning approach with unlabeled data. The purpose of a clustering task is to separate datapoints into a number of different groups and the datapoints within the same groups share some kinds of similarities based on the distance metrics we choose.

In our study, the original GPS data collected from the users is not labeled semantically, thus practitioners need to label the raw data first, to find the frequently visited places, which can be regarded as a clustering task in some sense. Here, frequently visited places refer to the places where the users visit frequently and stay for a relatively long time period. For example, a frequently visited place can be someone's home, his/her workplace or his/her school.

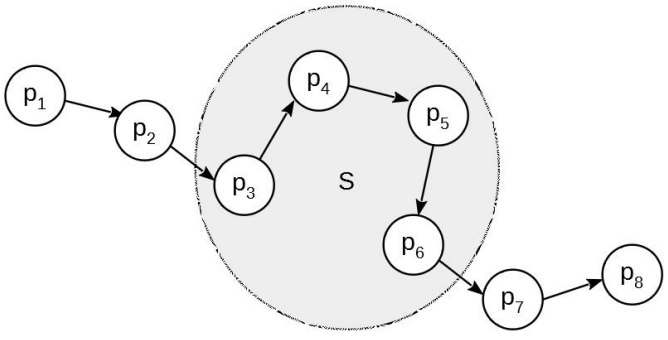


Figure 2.1: A GPS trajectory and a stay point.

Fig. 2.1 exhibits a GPS trajectory and its corresponding stay points. In this figure, $p_1 \sim p_8$

represent the GPS data points and the shaded area, S, denotes the stay point which contains 4 GPS datapoints, $p_3 \sim p_6$.

There are numerous machine learning methods which can be used for clustering GPS datapoints. For instance, K-means [Wu et al., 2008] is a widely used method for many clustering tasks. It measures the closeness between datapoints through calculating the **Euclidean** distances. The main advantage of K-means is that it is computationally efficient. But it cannot handle data with complex shapes and it is sensitive to noises because it uses Euclidean distances. If we want to find the frequently visited places, then we only care about the GPS data collected in the significant places and ignore the less important data. However, we cannot achieve this goal through using K-means because it cannot distinguish useful datapoints from noise datapoints. Moreover, it needs to set the cluster number properly in advance, otherwise the obtained result will not be as expected. However, we do not access to such prior knowledge in many cases.

One alternative is to use Gaussian Mixture Models (GMMs) [Reynolds, 2015]. GMMs are Probabilistic Graphical Models (PGMs). As opposed to K-means, in a GMM, each sub-Gaussian distribution represents a cluster and the category assignments of the datapoints depend on the corresponding likelihoods. GMMs are usually solved by the Expectation Maximization (EM) algorithm [Moon, 1996]. Like K-means, GMMs also need the prior knowledge of the cluster number. However, it is not acquirable in many real-world cases. In [Cho, 2016], the author modified the standard GMM to make the algorithm more robust to noise so as to cluster the original GPS datapoints. In addition, by using the Dirichlet process prior [Neal, 2000], GMMs can be even further developed as a nonparametric hierarchical model, called Hierarchical Dirichlet Process (HDP) [Teh et al., 2005], in which the number of sub-Gaussian models does not need to be specified in advance. In [McInerney et al., 2013], the authors developed a location HDP-based approach to model heterogeneous location habits and tackled with the data sparsity issue successfully.

Another feasible method is called Density-based Spatial Clustering of Applications with Noise (DBSCAN) [Ester et al., 1996], which is a density-based clustering algorithm. DBSCAN recognizes the core areas by setting the minimum points number and neighbourhood radius. In contrast with K-means, DBSCAN can handle data with complex shapes and is robust to noise. Moreover, it does not require to know the number of cluster in ad-

vance. However, it still needs to set some parameters properly, i.e., the minimum points number and the neighbourhood radius, to recognize the core areas and it treats the non-core data points as noise. Due to this reason, DBSCAN is not suitable for clustering the GPS data that is unevenly distributed in space.

Fig. 2.2 illustrates how the DBSACN algorithm works, where Eps is the neighbourhood radius.

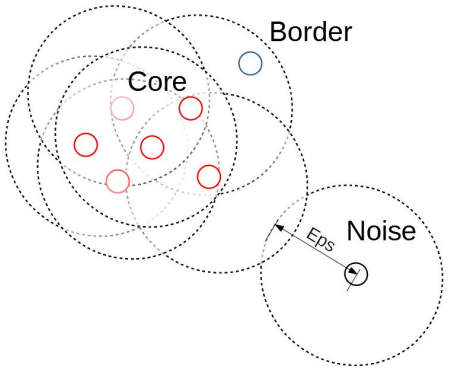


Figure 2.2: DBSCAN.

As for other methods proposed in literature, in [Do et al., 2012], the researchers proposed a grid clustering method to labeled GPS data. This grid clustering algorithm separates the GPS data via grids and it focuses on detecting the stay points within a set of square regions, while fails to reveal the mobility at a larger scale. Another possible approach is proposed by [Zheng et al., 2009], which is a hypertext induced topic search (HITS)-based inference model. It is proposed to mine interesting locations and travel sequences through using a GPS dataset of large scale in a certain region. In this model, especially, the travel interests and the travel experiences of the users are taken into account. In the work of [Zheng et al., 2010a], the authors took advantage of a real-world GPS dataset collected from more than 150 users over a time period of 2.5 years, to devise a location recommendation model. This model is able to discover both the interesting locations and possible activities.

The comparisons of different clustering methods are summarised in Table 2.1.

Table 2.1: Comparisons of Different Clustering Methods

Method	Distance metrics	Parameter
K-means	Euclidean	Cluster number
GMM	Log-likelihood	Cluster number
DBSCAN	Density	Minimum points, radius
HDP	Log-likelihood	Concentration parameter
HITS	Euclidean	Cluster number, hierarchy number

2.1.2/ Clustering GPS Trajectories

Although through discovering frequently visited places can reveal human mobility patterns in a sense, we attempt to shed some light on human mobility patterns in a more detailed manner. That is to say, we want to use not only the data collected from the frequently visited places but also all the user mobility data. Therefore, in this thesis, one of our goal is to cluster GPS trajectories so as to find the common patterns existing in GPS data. A applicable way to achieve this goal is to cluster GPS trajectories.

Researchers have developed many methods for clustering GPS trajectories [Bian et al., 2018], [Castro et al., 2013]. Some researchers used K-means [Jiang et al., 2012], [Ashbrook et al., 2003] in their work. However, K-means cannot handle the trajectories with complex data shapes or noise because its clustering metrics is based on the Euclidean distance. Besides, similar to Gaussian Mixture Models, it also needs the prior knowledge of cluster number as we mentioned before.

DBSCAN is capable of dealing with the data with arbitrary shapes, therefore it can be used to cluster GPS trajectories [Tang et al., 2015], [Yu et al., 2017]. However, it treats the non-core data points as noise so it cannot deal with unevenly distributed data. From our study, we argue that the trajectory parts with less data density are also essential to demonstrate the human mobility, thus DBSCAN is not suitable for our task. Dynamic Time Warping (DTW) is a sequence aligned approach that is able to measure the similarity between two different time series regardless of sequence lengths and time ordering [Agrawal et al., 1993]. However, when it is used to measures the similarity of two GPS trajectories, it can be easily affected by noise. Therefor DTW is not suitable for our task either.

In particular, some researchers focus on discovering the correlations between locations through the use of the user location history [Khetarpaul et al., 2011], [Zheng et al., 2011]. Furthermore, they utilized the travel experiences of the users and the correlations between the visited locations to construct a personalized location recommendation system. In the work of [Xiao et al., 2010], the researchers attempted to find the similar users by estimating the closeness of their GPS trajectories. To this end, first, they build the semantic location history (SLH), for instance, 'school' → 'bus stop' → 'home'. Then, they estimated the similarities between different users by using the maximal travel match (MTM) algorithm. [Lou et al., 2009] proposed a global map-matching method, ST-Matching algorithm. Compared to other methods, this algorithm considers both the spatial and topological structure of the road networks. In addition, it takes the speed and temporal constraint of the GPS trajectories into count. In the study of [Zheng et al., 2010b], the authors aimed to construct a user-specific recommendation system through estimating the correlations between different trajectories. In order to do so, they used the travel experiences of the users and the sequentiality of the locations.

However, the aforementioned methods have their limitations. Clustering real-world GPS trajectory data is a very tricky problem because, firstly, different trajectories may have different data lengths due to the data collecting mechanism; secondly, the shapes of the trajectories may be very complex and unevenly distibuted in space in some cases; thirdly, GPS data may contain noisy information. Therefore, in order to handle these problems, we need to devise a novel method to achieve our goal. In our work, we propose a probabilistic approach, in which we estimate the probability densities of the GPS trajectories first, then use the Kullback-Leibler divergences as the distance metrics to cluster the GPS trajectory data. By doing so, we can avoid the aforementioned issues successfully.

2.2/ PREDICTING NEXT USER LOCATION

Forecasting the next places that users will visit is an interesting research topic. It can be useful for many applications. For instance, it can be used for travel destination recommendation for tourists. Moreover, human behavior is highly related to locations, thereby we can improve the predicting accuracy by using the smartphone usage contextual information, e.g., temporal information, application usage, call logs and WiFi status, etc.

The next-place prediction can be classified into two groups of learning tasks. One is to predict the next time-slot location, the other kind is to predict the next visit location. The predicting task of next time-slot location is to predict the place where the users stay in the next-time slots.

2.2.1/ Machine Learning-based Prediction Methods

In [Baumann et al., 2018] and [Do et al., 2014], the authors applied various machine learning techniques to accomplish both the next time-slot location prediction and the next-place prediction. In particular, they exploited how different combinations of contextual features related to smartphone usage can affect the predicting accuracy. Meanwhile, they also compared the predicting performance of individual models and generic models. One class of the task is to predict the transitions among the places, i.e., the next visit location. In the task, different tasks are regarded as a set of separated places and the data related to these places can be either semantic labels or spatial coordinates.

In this thesis, in particular, we focus on predicting next visited places. This task can be regarded as a time series prediction task. A time series is a series of datapoints indexed in the order of time appearance. Since human behavior is stochastic by nature, thus deterministic may cause the overfitting issue. Overfitting is a phenomena that after we train a the model, the trained model fits the training data too well but fails to have good performance on the testing data. This may be due to the limited amount training data or noise in the training data. To overcome this difficulty, some probabilistic models, which leverage the conditional probabilities to make predictions, can be the alternative options. For the events A and B, the conditional probability $P(A|B)$ is defined as:

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{2.1}$$

where $P(A,B)$ is the joint probability and $P(B)$ is the marginal probability.

In the context of predicting the location of users, one can let B be some context events related to the location information, for instance, the hour of the day or the day of the week, and A be the next visit place. If we can calculate the marginal probability $P(B)$ and the joint probability $P(A,B)$, then we can leverage Eq. (2.1) to predict the next visit place via computing the conditional probability $P(A|B)$.

Based on this idea, in [Do et al., 2012], the researchers developed the contextual conditional models for both the next-place prediction and the visit duration prediction by calculating the conditional probabilities between contextual features. The duration model is based on mixed log-Normal distributions of relation contextual information. In order to increase the fidelity of the prediction, they developed a general model and personalized model.

In [Do et al., 2015] and [Peddemors et al., 2010], the researchers presented the probabilistic prediction frameworks based on Kernel Density Estimation (KDE) [Davis et al., 2011]. KDE is a non-parametric method in statistics to estimate Probability Density Function (PDF). KDE casts the problem of PDF estimation into data smoothing task and one of the key issue is choosing the proper bandwidth. [Do et al., 2015] utilized conditional KDE to predict the mobility events and [Peddemors et al., 2010] devised a set of ad hoc kernels for different context information types. Additionally, [Scellato et al., 2011] proposed to use nonlinear time series analysis of the arrival time and residence time for location prediction.

Various machine learning models have been applied to next-place prediction, such as Naive Bayes (NB) [Muhlenbrock et al., 2004], Markov models [Yu et al., 2017], Hidden Markov Models (HMMs) [Cho, 2016], Dynamic Bayesian Networks (DBNs) [Etter et al., 2013], [Patterson et al., 2003], etc. These models attempt to forecast the future states of human behavior by computing the state transition probabilities. Nevertheless, these methods have their disadvantages, when the number of possible states expands, the calculation load will grow exponentially.

[Liao et al., 2007] introduced a hierarchical Markov model which can model a the daily moments of users in urban areas. This model utilizes not only raw GPS data but also semantic information, such as different transportation modes. The author also used the Rao–Blackwellized particle filter [Doucet et al., 2013] to improve inference efficiencies.

Bayesian Networks [Heckerman, 2008] are a kind of Probabilistic Graphical Models (PGMs). Bayesian network represents the variables and the dependencies between them by using Directed Acyclic Graph (DAG). Normally, devising a Bayesian Network requires domain knowledge from experts, which is not always easy to do.

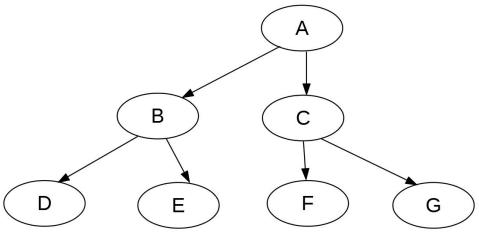


Figure 2.3: The structure of a Bayesian Network.

The model structure of a classic Bayesian network is illustrated in Fig. 2.3, where $A \sim G$ represent different variables and the arrows denote the dependencies between the variables.

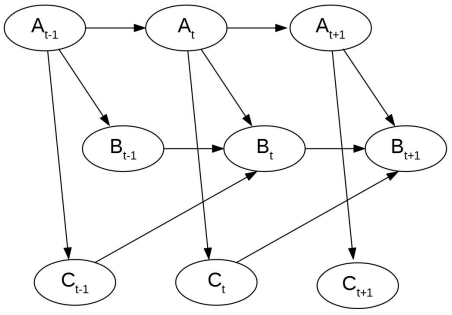


Figure 2.4: The structure of a Dynamic Bayesian Network.

The model structure of a Dynamic Bayesian Network is illustrated in Fig. 2.4, where t is the time point, A, B and C represent different variables and as opposed to classic Bayesian networks, the dependencies between variables are related to time in the Dynamic Bayesian Network.

However, the aforementioned conventional machine learning models are not applicable to WiFi fingerprint-based user location prediction. Because to represent the time series state transition, due to the complex relationship between the WiFi RSSI values and the coordinates, neither Kalman filter-based approaches [Yang et al., 2019], Bayesian network-based model or hidden Markov model-based approaches [Krogh et al., 2001] are suitable for the tasks.

2.2.2/ Deep Learning-based Prediction Methods

In order to solve this problem, we can resort to deep learning techniques, for instance, Recurrent Neural Networks (RNNs) [Elman, 1990]. The RNN is a widely used deep learning model specializing in time series prediction. The unfold structure of a RNN is depicted as in Fig. 2.5. The model structure of a RNN consists of the input layer, the hidden layer and the output layer as shown in Fig. 2.5. RNNs have other two variants, the Long Short-Term Memory (LSTM) [Gers et al., 1999] and Gated Recurrent Units (GRUs) [Chung et al., 2014]. LSTM solves the dependency problem in RNNs using a special structure. A LSTM unit has three gates, namely, an input gate, an output gate and a forget gate. These gates regulate the cell states of the LSTM. The GRU adopts a lighter structure compared to the LSTM while they has similar performance. In [Hoang et al., 2019], the authors compared different types of Recurrent Neural Networks including the vanilla RNN model, the Long Short-Term Memory model, the Gated Recurrent Unit model and the bidirectional LSTM [Graves et al., 2005] for accurate RSSI indoor localization. They also employed a weighted filter for both input and output layers to enhance the sequential modeling accuracy.

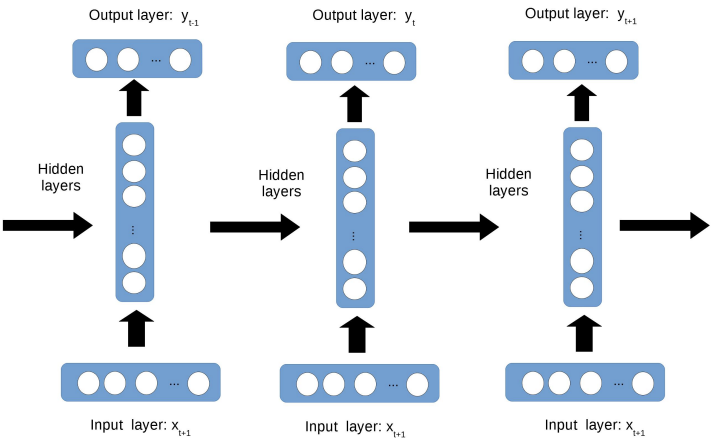


Figure 2.5: The architecture of a Recurrent Neural Network.

2.3/ INDOOR USER LOCATION RECOGNITION

Though GPS can provide the accurate information of user position, the disadvantage of GPS data-based methods is that GPS modules only function outdoors. Therefore, in indoor environment, we need to utilize WiFi fingerprint-based approaches as an alternative to study human mobility (other methods, for instance, laser-based methods, could be the options as well, however, in this thesis we only focus on smartphone usage data). WiFi fingerprint data records the Received Signal Strength Indicator (RSSI) values, which are numerical vectors related to the actual geographic location of smartphone users. Since WiFi fingerprints cannot directly be used to localize smartphone users, it needs to be labeled manually. The label values can be the building IDs, floor IDs or GPS coordinates.

In the literature, researchers have explored various types of machine learning techniques, both conventional machine learning and deep learning methods, on location recognition and prediction with WiFi fingerprint data. There are different kinds of research tasks related to WiFi fingerprints. Some researchers used WiFi fingerprints to identify building IDs and floor IDs, which can be regarded as classification tasks.

2.3.1/ CLASSIFICATION-BASED LOCATION RECOGNITION

For classification tasks, WiFi fingerprint data is labeled with building IDs and Floor IDs. Many conventional machine learning algorithm can be used for this type of classification task, for example, Decision Tree (DT) [Wu et al., 2008], K-nearest neighbors (KNN) [Bozkurt et al., 2015], Naive Bayes (NB) [Wu et al., 2008], Neural Networks (NNs) [Nowicki et al., 2017], Support Vector Machine (SVM) [Cortes et al., 1995], etc.

One widely used method is called the Classification and Regression Tree (CART) [Loh, 2011]. It is a kind of decision tree model, which can be used for both classification and regression tasks. CART is prone to be overfitting in practice. Therefore, in order to improve the performance, the CART method can be used as the basic estimators for the bagging method [Breiman, 1996] or the boosting method [Zhou, 2012]. For instance, Random Forest (RF) is a frequently used ensemble method called bagging [Breiman, 1996]. To overcome the overfitting problem, Random Forests utilize the bootstrap aggregating technique to decrease the variance of each decision tree.

Support Vector Machine (SVM) [Cortes et al., 1995] is another popular method. SVM is a kernel method which can be applied for both classification and regression. In a classification problem, SVM aims to find the optimal hyperplane to separate data.

In the work of [Bozkurt et al., 2015], the authors compared many traditional machine learning methods, for classifying buildings, floors and regions. In [Cramariuc et al., 2016], the authors clustered the 3D coordinates data by K-means and clustered the RSSI data by the affinity clustering algorithm, respectively.

Since WiFi fingerprint data are usually high dimensional, some deep learning techniques can be utilized for dimension reduction. Dimension reduction is to transform data from a high dimensional space to a rather low dimensional space while the information in the data retains. Because using high dimensional data to train models directly may be too computationally expensive and easy to be overfitting. In [Nowicki et al., 2017], [Kim et al., 2018], the authors used auto-encoders to reduce the input dimension before using a Multi-Layer Perceptron (MLP) to classify buildings and floors.

One essential issue of using WiFi fingerprint data is to deal with the high dimensionality issue. Therefore, in some tasks, training a model to predicting the targets through using the learnt low latent representation is more accurate than using the original input data. In order to reduce the dimension, some deep learning-based dimension-reduction methods like Autoencoders (AEs) can be an appropriate choice [Nowicki et al., 2017], [Song et al., 2019], [Kim et al., 2018]. Autoencoders (AEs) [Hinton et al., 2006] are unsupervised deep learning models used to compress input data. An Autoencoder contains an encoder and a decoder as shown in Fig. 2.6. During the training process, the encoder aims to learn a low-dimension representation of the input while the decoder is to reconstruct the original input. After the training process, we can obtain the latent representation of the input. Therefore, Autoencoders are commonly used for dimension reduction.

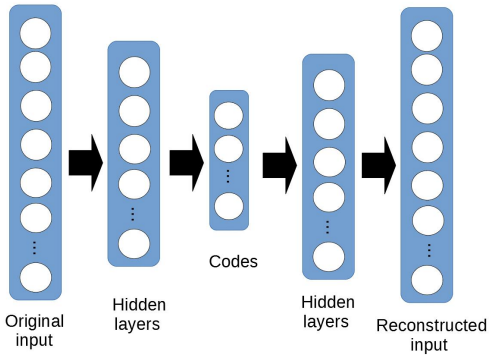


Figure 2.6: The architecture of an Autoencoder.

2.3.2/ ACCURATE LOCATION RECOGNITION

However, treating user the location recognition problem as classification tasks is only able to localize users at the accuracy level of buildings or floors. In some cases, we need to know the exact coordinates of users to proceed some tasks, for example, indoor navigation. In this case, we ought to frame location recognition as a regression problem. It means that we use WiFi RSSI vectors as the learning input and use the actual user location coordinates as learning targets of the proposed models.

Conventional machine learning models such as, Decision Tree, K-nearest neighbors and Random Forests can be used for such regression tasks. Specially, in [Torres-Sospedra et al., 2015], the researchers compared 51 different distance metrics to investigate the most suitable distance functions for accurate WiFi-based indoor localization. Some researchers used Gaussian Processes (GPs) to model the relations between the WiFi signal strengths and the corresponding indoor locations [Ferris et al., 2007], [Hähnel et al., 2006], [Yiu et al., 2015]. The Gaussian process is a type of stochastic process. The GP uses the kernel methods to measure the similarity between datapoints and to predict new values. The main drawback of the GP is that it is highly computationally expensive thus it is not suitable for datasets with large scales.

Besides, the aforementioned conventional machine learning models, we can also adopt advanced deep learning methods to slove the WiFi fingerprint-based location recognition problems. For accurate positioning, namely, interpreting WiFi RSSI values into actual numerical coordinates, the main issue of using conventional neural networks is overfitting.

For a traditional neural network, once it is trained, it can be regarded as a deterministic model (even the training process is stochastic). The neural network can be described as follow:

$$y = \mathcal{F}(x; w) \tag{2.2}$$

where x and y are the input and output of the NN, respectively, $\mathcal{F}$ represents the neural network structure and w are the weights of the NN.

Accordingly, the training loss (for instance, typically, mean squared errors) of NNs can be described as follow:

$$Loss = \frac{1}{N} \sum_{n=1}^{N} (\hat{y_n} - y_n)^2 \tag{2.3}$$

where N is the total number of the input, $\hat{y}$ is the model target and n is the mini batch size.

In the research of [Song et al., 2019], the authors used an auto-encoder network to reduce the data dimension, then used a CNN to proceed accurate user positioning. Deep learning methods, such as Convolutional Neural Networks (CNNs) [LeCun et al., 1998], Autoencoders (AEs) [Hinton et al., 2006] and Recurrent Neural Networks (RNNs) also have been utilized in WiFi fingerprint data-based positioning tasks. For instance, [Ibrahim et al., 2018] used a CNN model for time-series analysis. Generally, in order to provide good wireless Internet connection, modern buildings have many different WiFi access points, thus RSSI value data in many situations, can be very high dimensional. Due to this reason, it is reasonable to reduce the data dimension before carrying out a regression or classification task using WiFi fingerprints.

Principal Component Analysis (PCA) [Abdi et al., 2010] is a dimension reduction technique. PCA calculates the correlation matrix of original input data first, and then proceeds eigenvalue decomposition on the correlation matrix. However, in our case, each feature of the WiFi RSSI data has the equal contribution to the output, therefore PCA is not suitable for our problem.

Convolutional Neural Networks are a kind of deep neural networks that are widely used for images analysing, signal processing and natural language processing. The CNN includes various operations such as convolution operation, pooling operation and flatting operation. Each input channel of the CNN represents different colors of images. Convolution operation is to use a filter, which is a matrix, to detect the features of images. The

size of the matrix is called the kernel size. The stride is the shift length of the kernel in the convolution operation. After convolution operation is pooling operation, which aims to reduce the dimension of the convolved features and find the dominant features. There are two types of pooling, max pooling and average pooling. The difference is that max pooling is to return the maximum values of the convolved features while average pooling is to return the average values of all convolved features. After the pooling is to use a flatten layer to connect a MLP, for example, a classifier or a regressor. The model structure of a typical CNN is illustrated in Fig. 2.7.

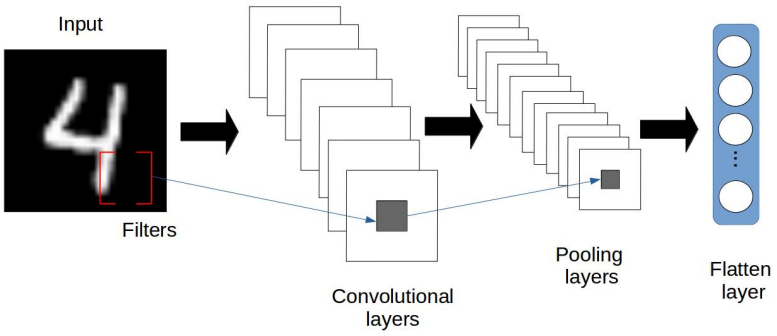


Figure 2.7: The architecture of a Convolutional Neural Network.

In many situations, a NN model is powerful enough to obtain satisfying results. However, in some cases, for instance, to solve a high non-Gaussian inverse problem (which means that a input value may correspond to multiple possible output values), using a traditional deterministic neural networks will lead to very poor modeling results [Bishop, 2006]. A good solution to this issue is to seek for a probabilistic framework that can calculate conditional probability distributions between input and output.

A class of probabilistic methods is called Maximum Likelihood Estimation (MLE), which uses likelihood as the optimization objective. The MLE methods are flexible but prone to be overfitting, especially when it comes to the cases in which the data are sparse or noisy.

$$p(\theta|D) \propto p(D|\theta) \tag{2.4}$$

where D is the dataset and θ is the model parameters. $p(\theta|D)$ is the posterior and $p(D|\theta)$ is the likelihood.

Mixture Density Networks (MDNs) are deep learning models using the Maximum Likelihood Estimation method [Bishop, Christopher M, 1994]. In a MDN, the final output is sampled by a mixture distribution rather than computed directly as opposed to conventional neural networks. One advantage of MDNs is that they can be applied to an estimation situation in which a large variety lies. For instance, we can incorporate more mixture modles of Gaussians to a MDN to enhance its estimating capacity for more complex distributions. However, as a MLE approach, MDNs also have obvious disadvantages. First, it needs to set some hyper-parameters properly (i.e., the mixture number of for a MDN), otherwise, it may not provide the desirable results due the underfitting or overfitting issue. Moreover, MLE methods may be severely biased when the training sample number are small, thus MDNs are not suitable for some tasks, for instance, the supervised step in semi-supervised learning. In practice, we find that MDNs suffer from computational instability when the mixture number at the output layer is large as well.

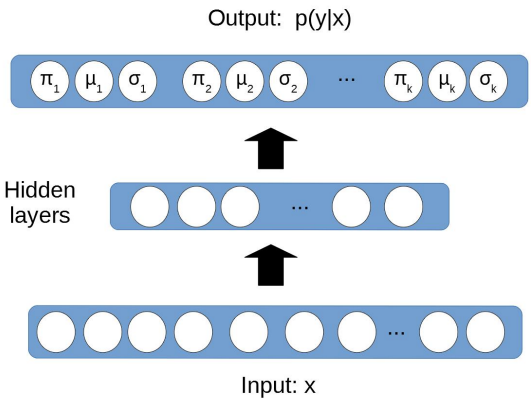


Figure 2.8: The architecture of a Mixture Density Network.

As demonstrated in Fig. 2.8, k is the mixture number, π_k is the weight of the Gaussian, μ_k is the mean and σ_k is the variance. In contrast to conventional neural networks, a Mixture Density Network deploys a mixed Gaussian distributions at the final output layer, so the MDN acquires the final output by sample from the mixed Gaussian distributions instead of using deterministic functions. This enables the MDN to solve the inverse-Gaussian problem.

In contrast with MLE methods, Maximum A Posteriori (MAP) methods consider not only the likelihood but also the prior of model parameters. MAP methods can be described as

follow:

$$p(\theta|D) \propto p(D|\theta)q(\theta) \tag{2.5}$$

where, $p(\theta|D)$ is the prior of the model parameters $q(\theta)$.

Compared to MLE, MAP is less easily to be overfitting and more robust to noise because it takes the prior of model parameters in count. MAP models can be solved by Variational Inference (VI) [Blei et al., 2017], [Zhang et al., 2018] or Markov chain Monte Carlo (MCMC) [Gilks et al., 1995].

Based on the idea of MAP, to alleviate the disadvantages of MDNs, Bayesian Neural Networks (BNNs), which apply Bayesian inference, have been introduced in [Hernández-Lobato et al., 2015]. BNNs follow the scheme of Maximum A Posterior estimation, in which the prior knowledge of model parameters and likelihood are both considered. As a result, MAP has the regularizing effect which can prevent overfitting. Unfortunately, in practice, we find that BNNs are not flexible enough for very complex distribution like our cases, i.e., recognizing user coordinates with WiFi fingerprints.

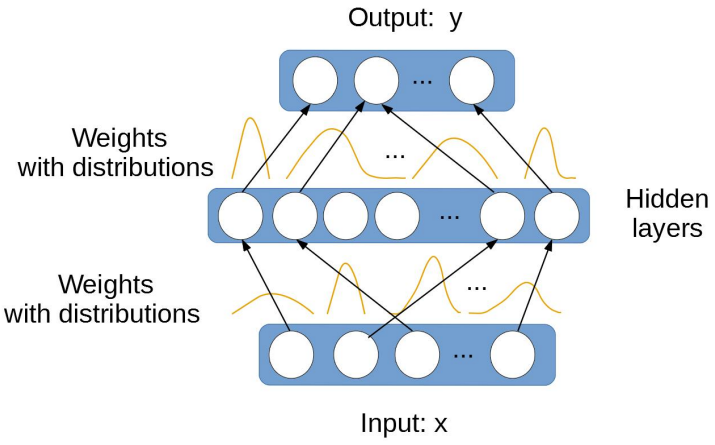


Figure 2.9: The architecture of a Bayesian Neural Network.

As for other MAP deep learning models, Variational Autoencoders (VAEs) [Kingma et al., 2013] are deep latent generative models trained in a unsupervised manner. Like conventional Autoencoders, a VAE consists of an encoder network and a decoder network. While the difference is that VAEs are designed to generative new image

samples and the latent variables of VAEs are stochastic. VAEs adopt two special techniques to infer the model parameters, one is called Variational Inference and the other is called the reparameterization trick. The structure of a VAE is depicted as in Fig. 2.10.

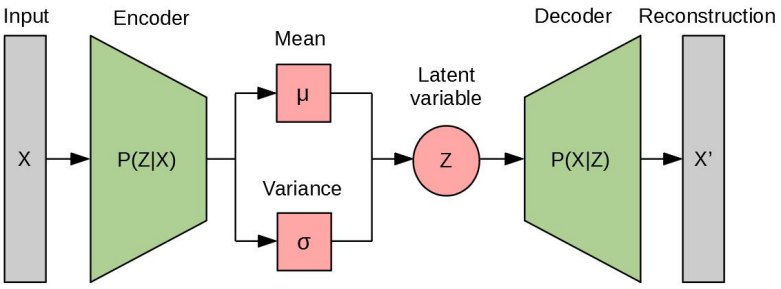


Figure 2.10: The architecture of a Variational Autoencoder.

Table 2.2: Comparisons of Different Deep Learning Models

Model	Learning Scheme	Learning Purpose
Autoencoders	Unsupervised	Dimension reduction
Convolutional Neural Networks	Supervised	Feature extraction
Mixture Density Networks	Supervised	Regression
Bayesian Neural Networks	Supervised	Regression/classification
Recurrent Neural Networks	Supervised	Sequential prediction
Variational Autoencoders	Unsupervised	Data generation

Table 2.2 summarises the differences between the popularly used deep learning models.

As we explained before, the data used for our study are GPS coordinate data and WiFi fingerprint data. These data commonly have the issues of sparsity, noise and high dimensionality. If we use conventional methods, it will result in poor modelling performance. For this reason, in this thesis, we propose a series of probabilistic methods to solve the aforementioned problems. First, for clustering GPS trajectories, we devise a novel method using Dirichlet Process Gaussian Mixture Models to estimate the probability densities of the trajectories instead of computing the Euclidean distances. Second, to predict the next indoor location, we design a deep learning model combining a CNN sub-model, a RNN

sub-model and a MDN sub-model, which allows us to detect the high dimensional features and avoid overfitting. Third, in order to recognize accurate indoor user location, we suggest that, compared to use the WiFi fingerprint data directly, it is better to use a representation of the input data. Based on this idea, we devise a Variational Autoencoder-based semi-supervised learning model and a Variational Information Bottleneck-based model, respectively. In the following chapters, the proposed methods in our work will be introduced.

3

Discovering Daily Outdoor Mobility Patterns

3.1/ Introduction

In this chapter, we focus on discovering the daily mobility patterns of multiple users in a specific region. Our purpose is to discover the mobility patterns for each individual based on their GPS location data. In order to do so, we need to cluster the daily trajectories of the users.

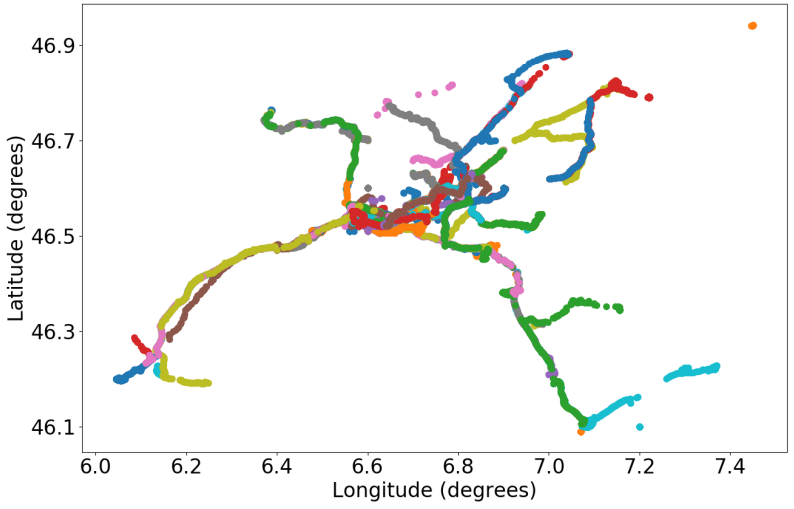


Figure 3.1: GPS data collected from a randomly selected user, different colors represent different data-collecting days.

As shown in Fig. 3.1, the mobility patterns of one individual consists of many different tra-

jectories (this data is from the MDC dataset [Kiukkonen et al., 2010], [Laurila et al., 2012], the detailed data description will be presented in the later section).

We hypothesize that the daily mobility of a user is rather regular and there are common mobility patterns shared among different daily trajectories. Generally, one may follow the regular daily itineraries, for instance, home → work place/school → home. Yet, on different days, the daily itineraries may not be the same. For instance, on the way to home, one may take a detour to do shopping in a supermarket. Hence, the objective of our research is to discover all the potential daily mobility from the data with the location information.

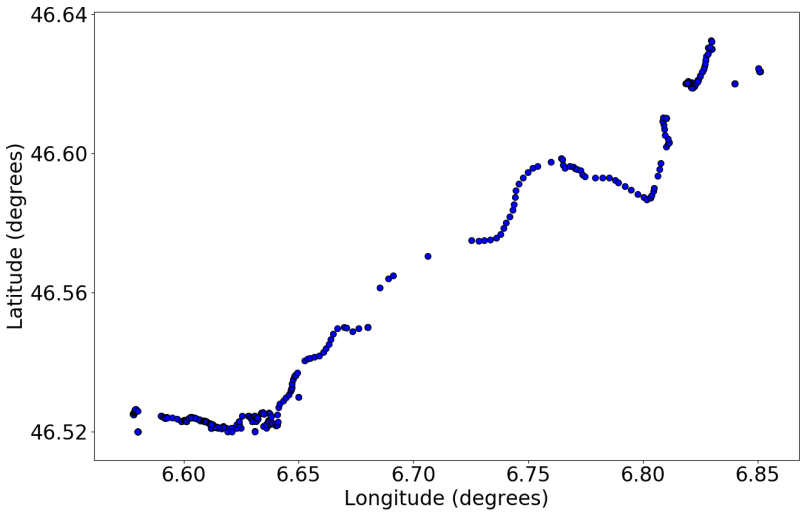


Figure 3.2: One randomly selected daily trajectory from a user.

We extract each day's trajectory from the whole dataset as shown in Fig. 3.2. It reveals that a daily trajectories recorded by GPS data is not distributed evenly in space, and is even not continuous in some areas. It may be caused by the data collecting procedure: some data collecting time range is actually relatively short (less than 24 hours, in fact, only few hours in some occasions), which leads to the data sparsity problem.

In order to overcome the data sparsity issue and to exploit as much information as possible from the available GPS data, we argue that a reasonable way to describe the daily trajectories is to estimate the probability density of the location data. And the relationships among the trajectories can be represented by their probability densities. As a result, we can discover all the mobility patterns for each user.

The tasks in this chapter are summarized as follows:

- Task 1: Estimating the probability density of each day. We will compare the results of the Gaussian Mixture Model (GMM) and the Dirichlet Process Gaussian Mixture Model (DPGMM) [Rasmussen, 2000];
- Task 2: Measuring the closeness between different trajectories. We will use the KL divergences as the metrics;
- Task 3: Discovering the similar mobility patterns among all the recorded daily trajectories. This can be regarded as a clustering problem;
- Task 4: Comparing the DPGMM-based algorithm with the GMM-based algorithms;
- Task 5: Identifying the minimum data length for discovering all the mobility patterns.

3.2/ METHOD

From Fig. 3.1 and Fig. 3.2, we can see that the GPS location data points are randomly spatially distributed. Besides, the distribution of each day consists of unknown number of heterogeneous sub-distributions. Therefore, it is reasonable to adopt the mixed Gaussian models to estimate the probability densities of daily mobility. The proposed clustering algorithm is summarized as follows:

- First, we estimate the probability densities of the trajectories via the Dirichlet Process Gaussian Mixture Models [Rasmussen, 2000];
- Second, we use the Kullback-Leibler (KL) divergence (computed via Monte Carlo sampling) as the distance metrics;
- Finally, we propose an automatic clustering algorithm based on DPGMM and KL divergence.

3.2.1/ PROBABILITY ESTIMATION

Since the daily trajectories are composed of different geo-locations, such as roads, homes, schools and offices, we need to use different sub-models to represent these

geo-locations. One feasible way is to combine a set of sub-models.

3.2.1.1/ GAUSSIAN MIXTURE MODELS

Gaussian Mixture Models (GMMs) are composed of a fixed number K of sub-components. The probability distribution of a GMM can be described as follow:

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\theta_k) \tag{3.1}$$

where x is the observable variable, π_k is the assignment probability for each sub-model, with $\sum_{k=1}^{K} \pi_k = 1,\ (0 \leq \pi_k \leq 1)$, θ_k is the internal parameters of the base distribution.

Let z_n be the latent variables to indicate the category assignment of the sub-models, then

$$z_n \sim \text{Categorical}(z_n|\pi) \quad \sum_{k=1}^{K} z_{nk} = 1 \tag{3.2}$$

where $z_n = \{z_{n1}, z_{n2}, ..., z_{nk}, ..., z_{nK}\}$, in which only one element $z_{nk} = 1$; it means that x_n is related to θ_k.

If the base distribution is a Gaussian, then

$$P(x|\theta_k) = N(x|\mu_k, \Lambda_k^{-1}) \tag{3.3}$$

where μ_k is the mean vector and Λ_k is the precision matrix.

Therefore, an observable sample x_n can be drawn from a GMM according to

$$x_n \sim \prod_{k=1}^{K} N(x_n|\nu_k, \Lambda_k)^{z_{nk}} \tag{3.4}$$

As it is illustrated above, one crucial issue of GMM is to pre-define the number of components K. This is a tricky problem because the probability distribution for each day's mobility is not identical and we do not have such prior knowledge in practice. Hence, using the GMM models with fixed K is not suitable in our case.

3.2.1.2/ DIRICHLET PROCESS GAUSSIAN MIXTURE MODEL

Alternatively, we resort to the Dirichlet Process Gaussian Mixture Model (DPGMM) [Rasmussen, 2000] (which is also called the Infinite Gaussian Mixture Model). As compared to the standard Gaussian Mixture Model, by using a Dirichlet Process (DP) prior for the mixture number, DPGMM does not need to specify the number of components in advance. Fig. 3.3 presents the graphical structure of the Dirichlet Process Gaussian Mixture Model.

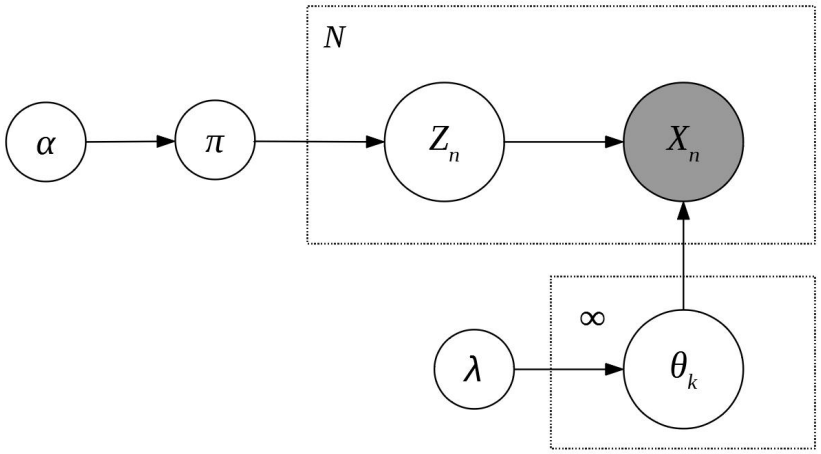


Figure 3.3: The plate representation of the Dirichlet Process Gaussian Mixture Model.

In Fig. 3.3, the nodes represent the random variables and especially, the shaded node is observable (the available dataset) and the unshaded nodes are unobservable (the latent variables); the edges represent the conditional dependencies between variables; the variables are within the plates means that they are drawn repeatedly. According to Fig. 3.3, the Dirichlet Process can be depicted as follow:

$$G \sim DP(\alpha,\ G_0) \tag{3.5}$$

G is a random measure, which consists of infinite base measure G_0 and λ is the hyper-parameter of G_0 (in our case, this is a set of Gaussian distributions), $\alpha \sim Gamma(1,\ 1)$ is the concentration parameter, N is the total sample number, θ_k is the parameters of base distribution, x_k is the observable data for θ_k, z_k is the latent variables that indicates the category of x_k.

G can also be explicitly depicted as follow:

$$G(\theta) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k} \tag{3.6}$$

where $\theta_k \sim G_0(\lambda)$, δ is the Dirac function, π_k determines the proportion weights of the clusters and the δ_{θ_k} is the prior of the θ_k to determine the location of clusters in space.

The Dirichlet Process can be implemented via two approaches, one is called the Chinese Restaurant Process (CRP) [Aldous, 1985] and the other is called the Stick-Breaking Process (SBP) [Sethuraman, 1994]. In practice, the Chinese Restaurant Process is more compatible with Markov chain Monte Carlo sampling method while the Stick-Breaking Process is more compatible with Variational Inference. More importantly, in terms of computational efficiency, Variational Inference is much faster than Markov chain Monte Carlo sampling.

Since our dataset is of large scale, we adopt the Stick-Breaking Process to implement the Dirichlet Process as the prior for π_k. The Stick-Breaking Process can be described as follow:

$$\pi_k = \beta_k \prod_{j=1}^{k-1} (1 - \beta_j) \quad k \geq 2 \tag{3.7}$$

where $\beta_k \sim \text{Beta}(1, \alpha)$.

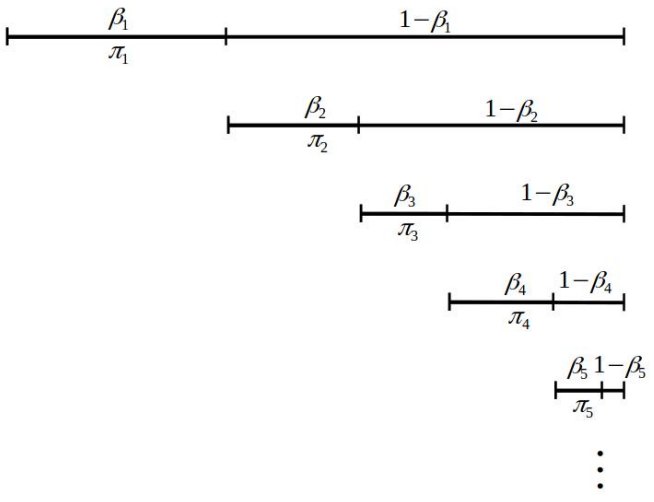


Figure 3.4: The Stick-Breaking Process.

Fig. 3.4 demonstrates the sampling scheme of the Stick-Breaking Process. In SBP, the "stick" breaks into different sub-parts with respect to different probabilities. When we want to create a new sub-model, we can break the remain of the "stick" and this process can be proceeded infinitely. As a result, we can obtain the infinite sub-mixture models.

Since $P(x|\theta)$ is Gaussian, $\theta = \{\mu, \Lambda\}$. μ is the mean and Λ is the variance. Further, let G_0 be a Gaussian-Wishart distribution [Rasmussen, 2000], then, $\mu_k, \Lambda_k \sim G_0(\mu, \Lambda)$. Therefore, similarly, we can draw an observable sample x_n from DPGMM:

$$x_n \sim \prod_{k=1}^{\infty} N(x_n|\nu_k, \Lambda_k^{-1})^{z_{nk}} \tag{3.8}$$

Variational Inference is used to solve the DPGMM models. As compared to Gibbs sampling, a type of Markov chain Monte Carlo (MCMC) method which consumes a large mount of calculating time, Variational Inference is relatively fast [Blei et al., 2006] especially when the dataset is large.

3.2.2/ COMPUTING KL DIVERGENCE

The Kullback-Leibler (KL) divergence is a metrics to evaluate the closeness between two distributions. For continuous variables, the KL divergence $D_{KL}(p\|q)$ is the expectation of the logarithmic difference between the p and q with respect to probability p and vice versa. From Eq. (3.9) and Eq. (3.10), it can be seen that the KL divergence is non-negative and asymmetric. Here, "asymmetric" means the distance from p to q is different from the distance from q to p unless they are identical distributions. In many occasions, the inequality of the KL divergence is notorious. However, in our method, on the contrary, we take advantage of the characteristics of inequality to reveal the similarities among different trajectories instead of using the symmetric metrics (for example, the Jensen-Shannon divergence [Fuglede et al., 2004]).

$$D_{KL}(p\|q) = \int_{-\infty}^{\infty} p(x) \log \left\{ \frac{p(x)}{q(x)} \right\} dx \tag{3.9}$$

$$D_{KL}(q\|p) = \int_{-\infty}^{\infty} q(x) \log \left\{ \frac{q(x)}{p(x)} \right\} dx \tag{3.10}$$

There is no closed form to compute the KL divergence by the definition of Eq. (3.9) and Eq. (3.10) for the Gaussian Mixture Models. Instead, we resort to the Monte Carlo simulation method proposed in [Hershey et al., 2007]. Then, the KL divergence $D_{KL}(p||q)$ can be calculated by:

$$D_{KL_{MC}}(p||q) \approx \frac{1}{n}\sum_{i=1}^{n}\log\left\{\frac{p(x_i)}{q(x_i)}\right\} \tag{3.11}$$

where n is the sample sample for the Monte Carlo sampling.

Similarly, the KL divergence $D_{KL}(q||p)$ can be calculated by:

$$D_{KL_{MC}}(q||p) \approx \frac{1}{n}\sum_{i=1}^{n}\log\left\{\frac{q(x_i)}{p(x_i)}\right\} \tag{3.12}$$

This method is to draw a rather large amount of i.i.d (independent and identically distributed) samples x_i from distribution p to calculate $D_{KL_{MC}}(p||q)$ according to Eq. (3.11) and $D_{KL_{MC}}(p||q)$ approximates $D_{KL}(p||q)$ as $n \to \infty$. It is the same for implementing Eq. (3.10) via Eq. (3.12). The results will be demonstrated in the later experiments. Furthermore, if we define a representative trajectory for a mobility pattern then we can identify whether a new trajectory belongs to this cluster by comparing it to the representative trajectory. To this end, we need to set a threshold with a lower bound and an upper bound for the KL divergence, then it can be used as the metrics to cluster mobility patterns.

3.2.3/ DPGMM-BASED TRAJECTORY CLUSTERING ALGORITHM

As mentioned before, our task is to find the trajectories which are mutually similar. For this reason, we treat the different mobility patterns as different clusters in which the daily trajectories are their sub-members. Even so, the trajectories within the same clusters still can not be treated as identically distributed as other conventional clustering methods because of different trajectory lengths. Hence, we need to devise an algorithm that is able to cluster the trajectories based on the distribution similarity and the aforementioned KL divergence can be applicable as the closeness metrics. Note that due to the large data scale and the number of the potential clusters, a solution with high accuracy is not acquirable in some cases. Therefore, instead of pursuing a very accurate result, our purpose is to obtain a relative accurate result in a reasonable amount of calculating time.

Algorithm 1 Mobility Pattern Discovering Algorithm

Input: X

Output: M

1: $P \leftarrow$ DPGMM(X) ▷ probability density estimation
2: Initialize:$M = \{M_k\}$ ▷ create the mobility patterns set
3: **while** $P \neq \emptyset$ **do**
4: $X_s = X_1$ ▷ set the baseline mobility for M_k
5: $M_k = \{X_s\}$ ▷ create current pattern M_k
6: **for** $d = 2, \ldots, D$ **do**
7: $D_{KL} \leftarrow (P_s, P_d)$ ▷ measure similarity
8: **if** $(\min(D_{KL}) < Th[0])$ & $(\max(D_{KL}) < Th[1])$ **then** ▷ two patterns are similar
9: add P_d to M_k ▷ add new member
10: **if** $D_{KL}[0] > D_{KL}[1]$ **then**
11: $P_s \leftarrow P_d$ ▷ change the baseline mobility
12: **end if**
13: **end if**
14: **end for**
15: remove $P_d \in M_k$ from P ▷ current pattern is finished
16: create M_{k+1} ▷ find new mobility pattern
17: add M_{k+1} to M
18: **end while**
19: **return** M

Table 3.1: Variables Description

Variable	Domain	Description
d	$\{1, 2, \ldots, D\}$	Number of data collecting day
X	$\{X_1, X_2, \ldots, X_d, \ldots, X_D\}$	Total GPS data (longitudes, latitudes
P	$\{P_1, P_2, \ldots, P_d, \ldots, P_D\}$	Probability density for X
M	$\{M_1, M_2, \ldots, M_k, \ldots M_K\}$	Total mobility patterns
M_k	$\{X_{k1}, X_{k2}, \ldots, X_{kn}\}$	Discovered mobility pattern
Th	{lower bound, upper bound}	Threshold for distinguishing patterns
D_{KL}	$\{D_{KL}(p\|\|q), D_{KL}(q\|\|p)\}$	KL divergences

The proposed algorithm is shown in Algorithm 1 and its variables are described in Table 3.1. The first step of the clustering algorithm is to calculate the probability densities using the Dirichlet Process Gaussian Mixture Models. At this step, we create a list, in which the members are the probability densities of each trajectory. Then, the first cluster is created with one trajectory as its first member and it also will be compared with other trajectories.

Afterwards, we select another daily trajectory in the list and calculate the KL divergences, both $D_{KL}(p||q)$ and $D_{KL}(q||p)$. The new trajectory is added to the current cluster if the minimum and maximum of the KL-divergences are smaller than the lower bound and upper bound of the thresholds respectively at the same time. If the $D_{KL}(p||q)$ is smaller than $D_{KL}(q||p)$, the new trajectory becomes the benchmark for the current cluster. An alternative way to do this is to compute the probability density of the current cluster using all the data of the discovered trajectories, however, the calculation cots will be expensive.

This step will be repeated until all the trajectories belonging to the current cluster are discovered at the end of this iteration. Then, all the members of the current cluster are removed from iteration because, we assume that each trajectories can only be a member of one mobility pattern. At the start of a new iteration, a new cluster is created. The above steps will be repeated until the list of the trajectory probability densities is empty. Finally, all the mobility patterns are discovered.

It can be seen that our algorithm is designed to discover the latent mobility patterns automatically without the pre-knowledge of the numbers of the existing patterns.

3.3/ Experiments and Results

3.3.1/ Dataset Description

We use the Mobile Data Challenge (MDC) dataset [Kiukkonen et al., 2010], [Laurila et al., 2012] to validate our method. This dataset records comprehensive smartphone usage information with fine granularity of time. The participants of the MDC dataset are up to nearly 200 and the data collection campaign lasts more than 18 months. This abundant information thus can be used to investigate individual mobility patterns for our research.

In order to collect the individual location information, as compared to other methods, for instance, through stand-alone GPS devices, using GPS-equipped smartphones is a more practical way to have a larger group of participants without affecting their daily life.

In our study, we attempt to find the trajectories that belong to the same mobility patterns, thus we focus the spatial information of the GPS records, namely, the latitudes and longitudes and the time-stamps of the data are not considered. Meanwhile, since we consider not only the significant places but all location records, we use the unlabeled data without any semantic information.

3.3.2/ Experimental Setup

In the conducted experiments, we randomly select 20 users with sufficient data. Each user's data is segmented by the time range of one day. Fig. 3.5 demonstrates the number of data collecting days for each user. It can be seen that the data collecting days for most users are more than 200. With such amount of data, we believe that it is possible to discover the mobility patterns of individuals.

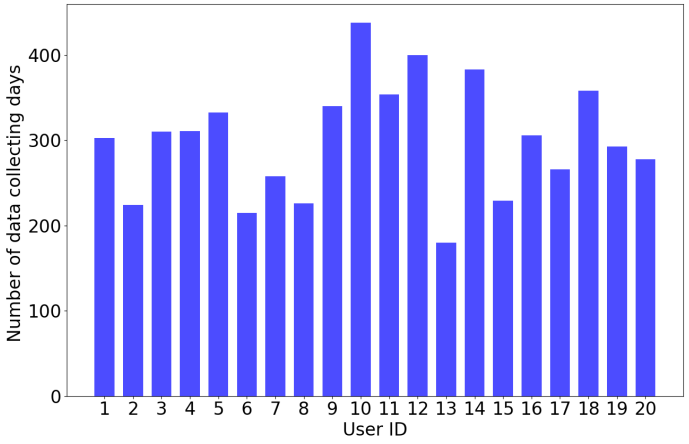


Figure 3.5. Number of data collecting days for each user.

However, as it is illustrated in Fig. 3.6 and Fig. 3.7, the data length of each day varies from less than 4 hours to 24 hours. Most of them are less than 8 hours. Hence, we should also be aware that some data may be missing because the GPS modules were turned off or were not functioning. Consequently, it is one of the reasons that causes the data sparsity problem. In the following sections, we will prove that our method can mitigate the impact of data sparsity.

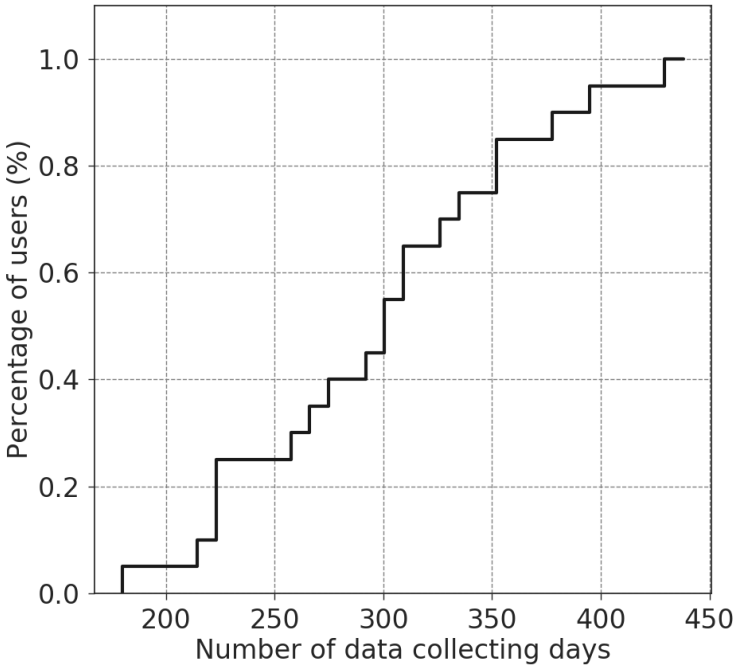


Figure 3.6. Empirical cumulative distribution of data collecting days.

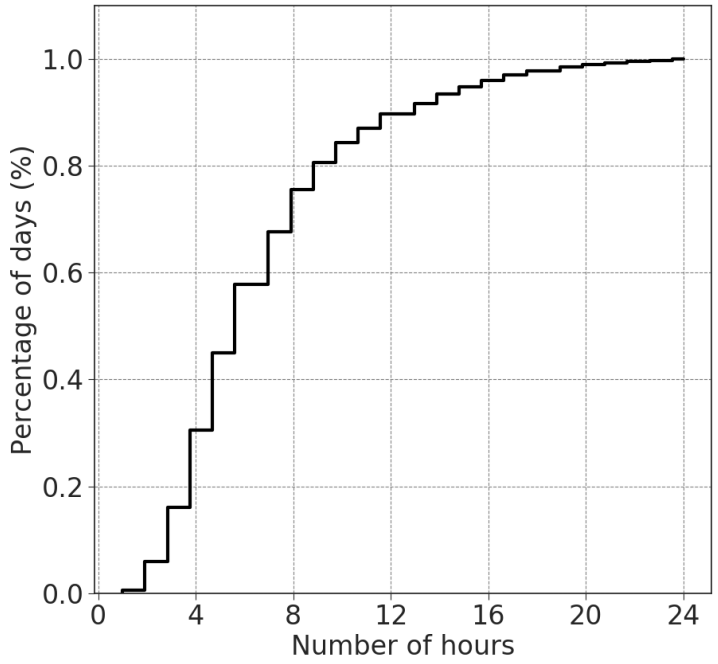


Figure 3.7. Empirical cumulative distribution of hours per data collecting day.

TABLE 3.2. DATA COLLECTING TIME.

	Average	Total
Collecting days for all users	300.25 days	6005.0 days
Collecting hours per day for all users	6.93 hours	41595.0 hours
Collecting hours per day for each user	6.67 hours	2084.65 hours

Table 3.2 summarizes the statistical temporal information about the GPS data for conducting the experiments. To test the performance of our method, we will conduct 5 experiments from different perspectives:

- We compare the DPGMM model with the GMM model on estimating the daily trajectories probability density;
- We use that the KL divergence to measure the closeness of different trajectories;
- We test our method on each selected user data so as to find the daily mobility patterns for each individual;
- We compare the results of the DPGMM models to a series of fixed-number component GMM models;

- We run the algorithm on the varying-length datasets so as to identify the minimum data length for discovering most mobility patterns of one individual.

3.3.3/ TASK 1: PROBABILITY DENSITY ESTIMATION

In this experiment, we compare the probability density estimation results of the GMM model and the DPGMM model.

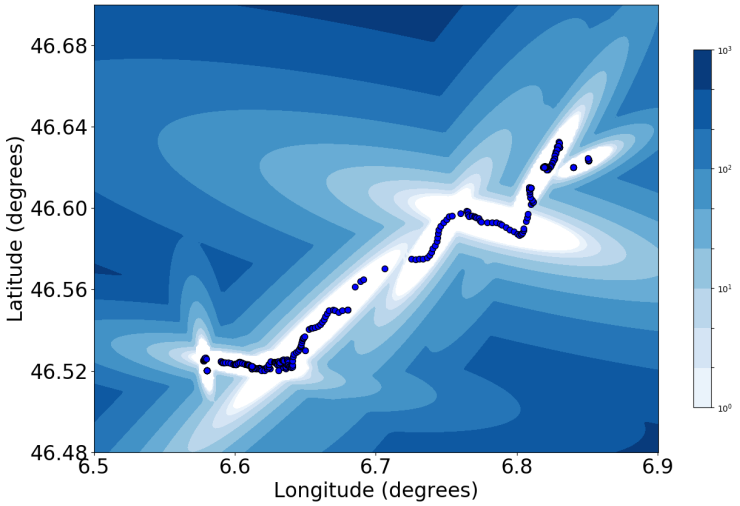


Figure 3.8. Distribution estimation by GMM (negative log-likelihood).

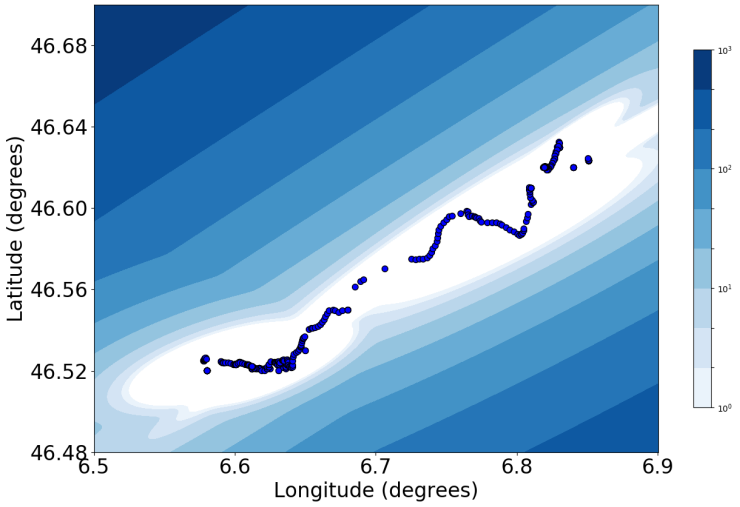


Figure 3.9. Distribution estimation by DPGMM (negative log-likelihood).

Fig. 3.8 and Fig. 3.9 show the density estimation results obtained by the GMM and the

DPGMM, respectively. It can be seen that, compared to the GMM model, the result of the DPGMM model is more smooth. It suggests that the DPGMM is not affected by the number of components and it infers more information from the original data and it is less influenced by data sparsity. That is to say, on the same dataset, the computational results of the DPGMM have higher fidelity. Hence in our approach, we chose the DPGMM to estimate the probability density of daily mobility.

3.3.4/ TASK 2: MEASURING DAILY TRAJECTORIES SIMILARITIES

In this experiment, we use the KD divergences as the metrics to measure the closeness between different trajectories.

As shown in Fig. 3.10, Fig. 3.11, Fig. 3.12 and Fig. 3.13, we select 5 daily trajectories from the data of one random user to present the KD divergences between different trajectories. The baseline trajectory is the Trajectory 1 and the rest of trajectories are chosen to make comparisons.

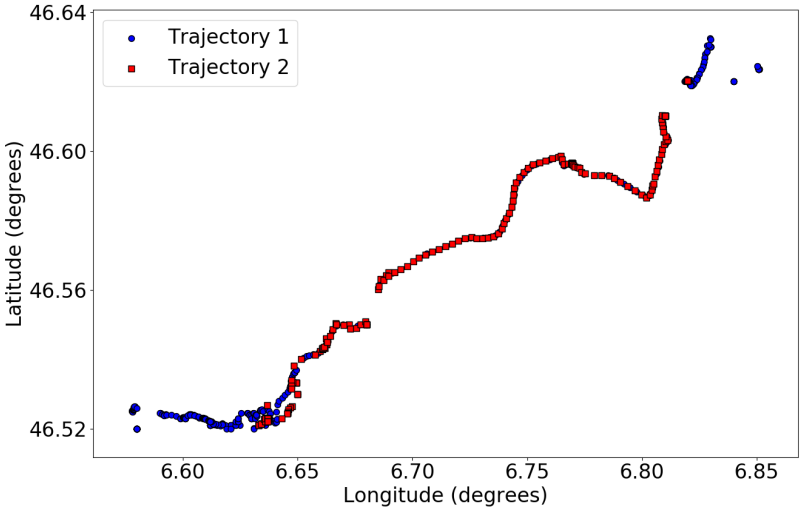


Figure 3.10. Trajectory 1 and Trajectory 2.

From Fig. 3.10, we can see that Trajectory 2 is nearly a subset of Trajectory 1 and thus $D_{KL}(p\|q)$ is larger than $D_{KL}(p\|q)$. Their values are both small, thus Trajectory 2 and Trajectory 1 can be regarded to belong to the same mobility pattern.

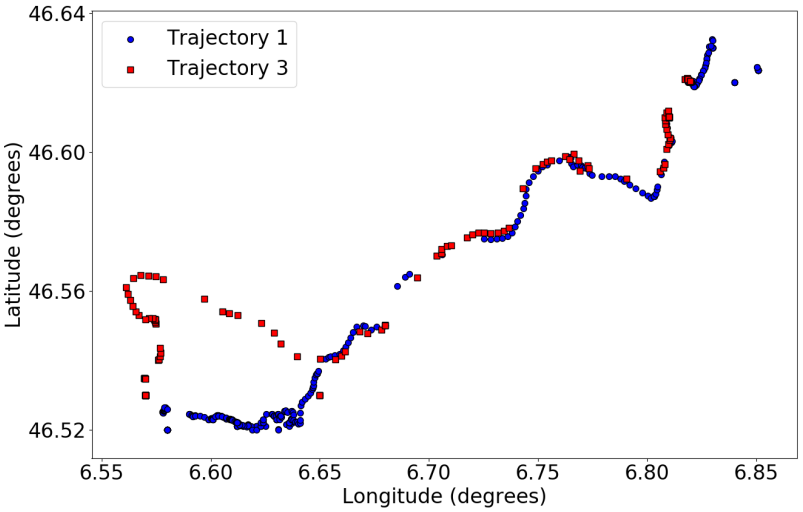


Figure 3.11. Trajectory 1 and Trajectory 3.

From Fig. 3.11, we can see that Trajectory 3 is very similar to Trajectory 1 and $D_{KL}(p||q)$ almost equals $D_{KL}(q||p)$. Hence, they also are the members of the same mobility pattern.

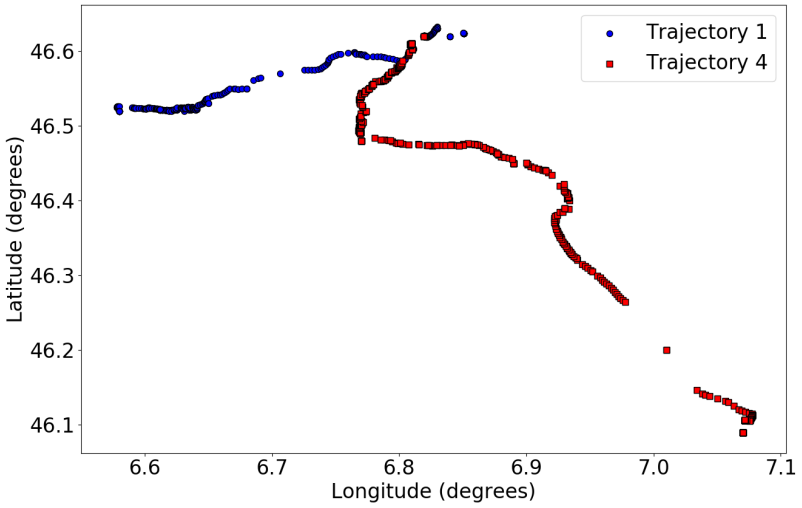


Figure 3.12. Trajectory 1 and Trajectory 4.

From Fig. 3.12, we can see that Trajectory 4 share a small part with Trajectory 1 whereas generally they are very different, thus $D_{KL}(p||q)$ and $D_{KL}(q||p)$ are both very large. Therefore, it is reasonable to recognize Trajectory 4 and Trajectory 1 as different patterns.

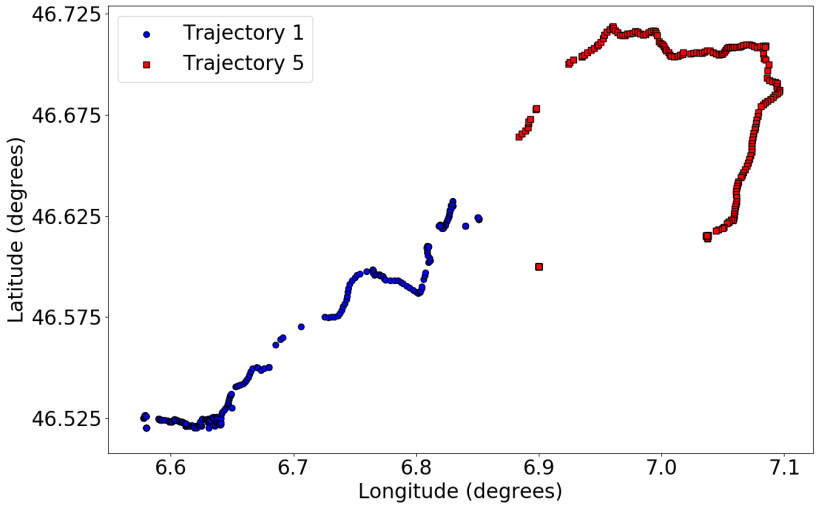


Figure 3.13. Trajectory 1 and Trajectory 5.

From Fig. 3.13, we can see that Trajectory 5 is totally different from Trajectory 1, thus $D_{KL}(p||q)$ is small but $D_{KL}(p||q)$ are very large. So they naturally are not in the same pattern.

TABLE 3.3. KL-DIVERGENCES BETWEEN DIFFERENT TRAJECTORIES.

p	q	$D_{KL}(p\|\|q)$	$D_{KL}(q\|\|p)$
Trajectory 1	Trajectory 2	7.21	2.82
Trajectory 1	Trajectory 3	1.28	1.83
Trajectory 1	Trajectory 4	19.07	1269.47
Trajectory 1	Trajectory 5	3.08	996.17

According to the trajectories in the Fig. 3.10, Fig. 3.11, Fig. 3.12 and Fig. 3.13 and the corresponding results in Table 3.3, it shows that the KL divergence is able to illustrate the differences among the trajectories and can be the metrics for clustering.

3.3.5/ TASK 3: DISCOVERING DAILY MOBILITY PATTERNS

In this experiment, we will use the proposed algorithm to discover the similar mobility patterns among all the recorded daily trajectories. We also will try to find how many patterns each user has and how many trajectories each pattern has.

3.3.5.1/ DISCOVERED PATTERNS

The partial results of the data from different randomly selected users are demonstrated in Fig. 3.14. It shows that, after clustered by our proposed algorithm, the data is split into different mobility patterns.

Each cluster is composed of trajectories close to each other even if they are not distributed with the same density in the space. This proves our methodology is able to find the different mobility patterns even under the condition of noise and discontinuous trajectories.

Fig. 3.15 shows that our methodology is not only able to identify the different patterns in the daily trajectories data but is also able to find the most representative trajectories for each mobility pattern.

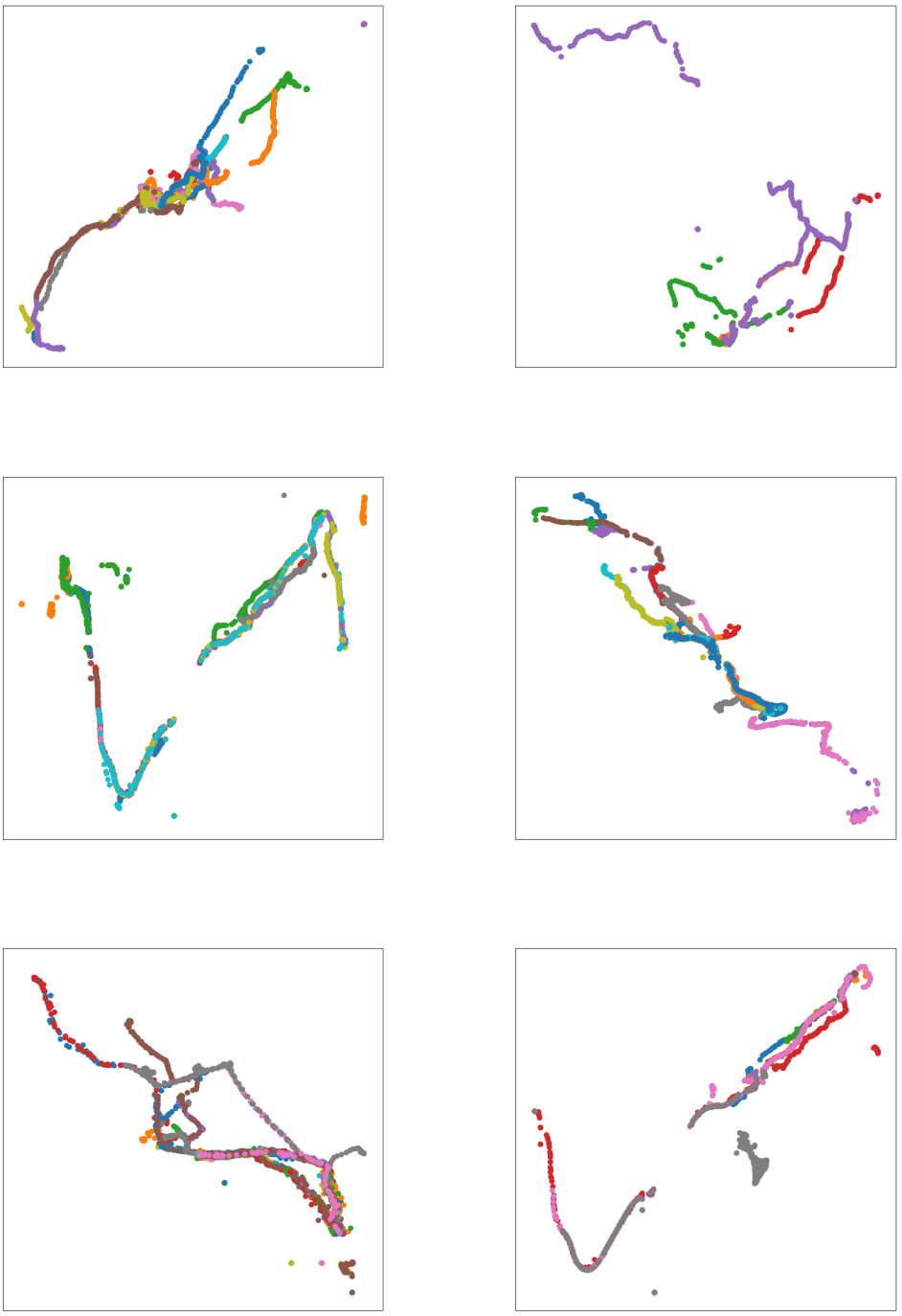


Figure 3.14. Discovered mobility patterns of three random selected users. Different colors denote different days.

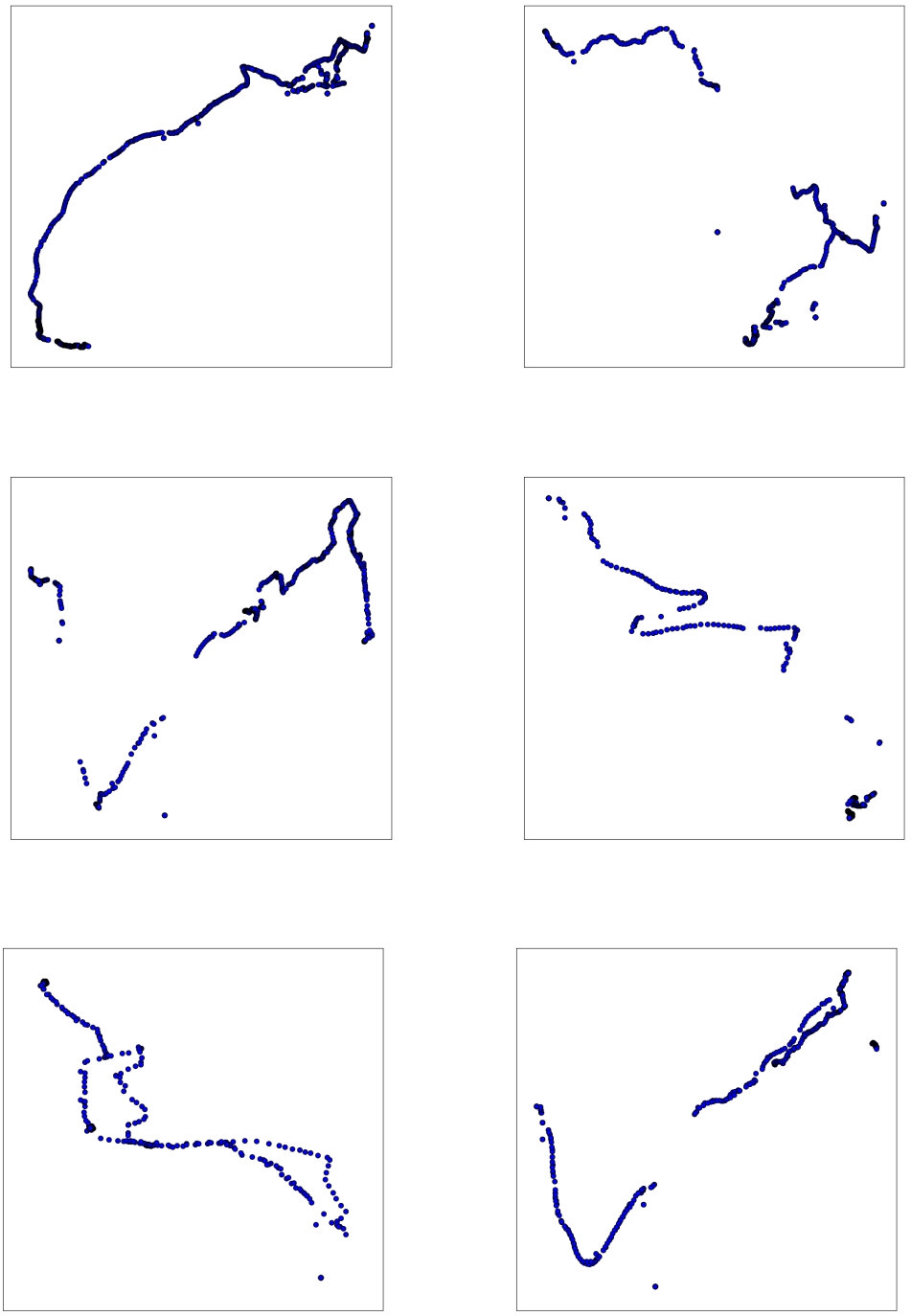


Figure 3.15. Representative trajectories for each discovered mobility patterns.

3.3.5.2/ NUMBER OF PATTERNS AND TRAJECTORIES

Fig. 3.16 shows the number of discovered mobility pattern for all the users in our experiments. We can see that the number of mobility patterns varies from 5 to more than 30 and most of them are about 10 to 15. It also can be found that the lengths of the data collecting days are not proportional to the number of discovered mobility patterns, which indicates that the results rely more on the individual behavior rather than the data lengths.

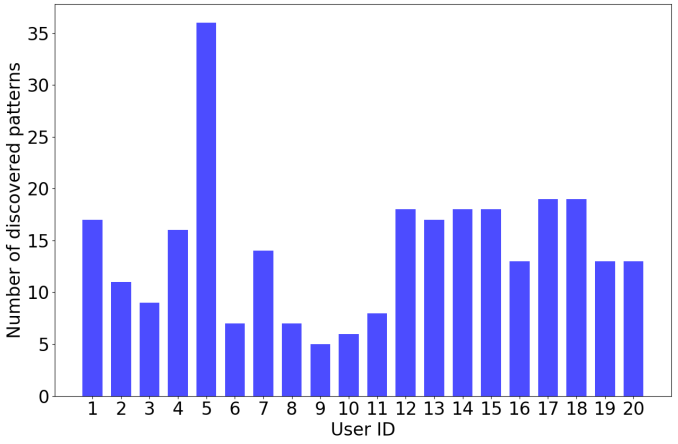


Figure 3.16. Number of discovered mobility patterns for each user.

3.3.5.3/ NUMBER OF MEMBERS FOR EACH PATTERNS

Fig. 3.17 depicts the number of members for each discovered mobility patterns for all users. We can see that most mobility patterns consist of less than 50 trajectories. Nearly 40% of the patterns have only one trajectory, whereas few patterns have more than 100 trajectories.

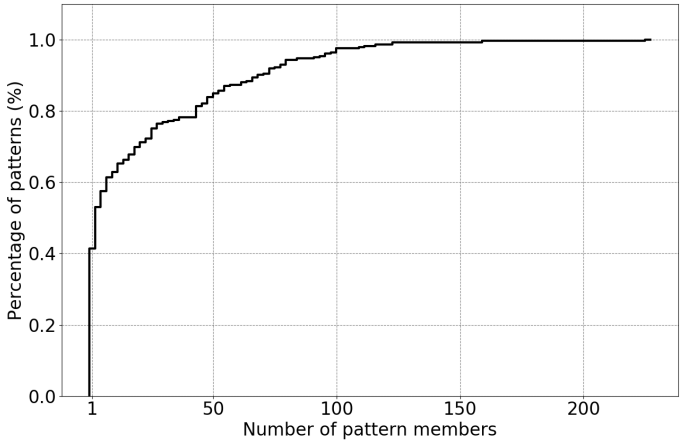


Figure 3.17. Empirical cumulative distribution of the members of the discovered patterns.

One should notice that the number of discovered patterns depends on the Kullback-Leibler divergence thresholds we set in the proposed clustering algorithm. When the thresholds are small, it means that the condition to be in the same mobility pattern is more strict and naturally the discovered mobility patterns are more and the members of each patterns are less, and vice versa.

3.3.6/ TASK 4: COMPARISON TO OTHER MODELS

In this experiment, to compare the Dirichlet Process Gaussian Mixture Models, we use a set of Gaussian Mixture Models with different numbers of components to estimate the daily mobility probability densities in our proposed clustering algorithm. The metrics we adopt to evaluate the results is the mean log-likelihoods. Higher log-likelihoods mean more reliable results.

TABLE 3.4. OVERALL MEAN LOG-LIKELIHOODS OF DIFFERENT MODELS

Model	Mean log-likelihood
KDE	-51991.03
GMM-1	-26078.15
GMM-2	-38514.32
GMM-3	-52431.62
GMM-4	-63794.70
GMM-5	-73508.10
GMM-8	-101306.32
DPGMM	**-24871.78**

The results shown in Table 3.4 indicate that changing the fixed number of component Gaussian Mixture Models can not enhance the clustering performance. On the contrary, the Dirichlet Process Gaussian Mixture Model can improve the clustering performance.

3.3.7/ TASK 5: VARYING DATA LENGTHS

In this experiment, in order to investigate how the data lengths, namely, the number of days of the data, affect the final results. We utilize different data lengths which varies from 50 days to 350 days. The obtained results are shown in Fig. 3.18.

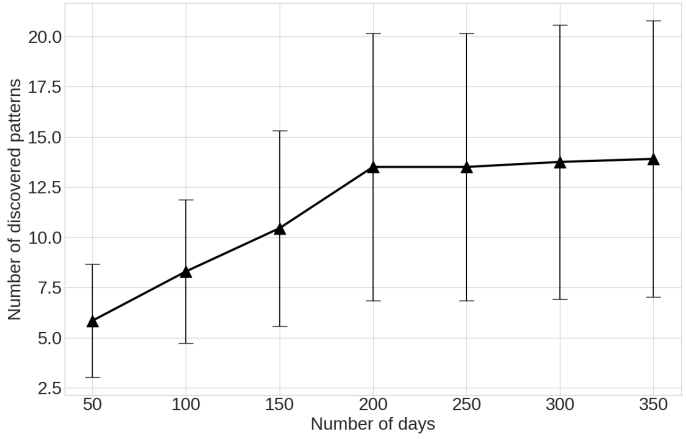


Figure 3.18. Average number of discovered patterns for different data collecting day length, error bars represent the standard deviations.

It can be seen that, from 50-day data length to 200-day data length, the average discovered mobility pattern numbers increase as the data length grows. While, when the data length is larger than 200 days, the patterns numbers change marginally. According to the results, we can say that, generally, a 200-day GPS dataset is large enough to discover most of the mobility patterns of an individual.

3.4/ CONCLUSION

In this chapter, we present a probabilistic approach to discover human daily mobility patterns based on GPS data collected by smartphones.

In our approach, human daily mobility is considered as sets of probability distributions.

The proposed approach consists of three steps. The first step is to estimate the probability densities of the GPS daily trajectories. We argue that the Dirichlet Process Gaussian Mixture Model is more appropriate than the standard Gaussian Mixture Model on this issue. This argument is validated by the corresponding experimental results. Further, in order to find the similar trajectories, one needs to measure the closeness between the trajectories. For this task, we choose the Kullback-Leibler divergence as the distance metrics. According to the computational results from the selected trajectories, we validate that the KL divergences are able to measure the similarities among the trajectories. Finally, we devised a novel automatic clustering algorithm combining the advantages of both DPGMM and the KL divergence so as to discover human daily mobility patterns without requiring the knowledge of the cluster numbers in advance.

For validation, we select the data of 20 random individuals from the MDC dataset to conduct a series of experiments. The results obtained show that our proposed approach can discern different mobility patterns and select the most representative trajectories for each mobility patterns from the GPS data. In addition, we also compare the DPGMM-based algorithm with a group of GMM-based algorithms with various fixed-number components, the results reveal that the DPGMM model performs better. Finally, testing our method on varying-length dataset leads to the results which suggest that a 200-day GPS is generally sufficient enough to discover most of the individual daily mobility patterns.

4

Predicting Indoor Location with WiFi Fingerprints

4.1/ Introduction

In the previous chapter, our work has focused on studying outdoor user mobility with using the collected GPS coordinate data. In this chapter, in order to have a comprehensive understanding of human mobility, we need to investigate user mobility from both indoor and outdoor aspects. However, GPS data-based approaches are not suitable anymore for studying indoor user mobility because GPS modules do not work well when smartphone users stay indoors. Therefore, in this chapter, we choose to use WiFi fingerprint data to study user mobility.

Our goal is to interpret the smartphone user location with the corresponding WiFi fingerprints. This task can be regarded as a high dimensional time-series regression task. The training input of our model are the RSSI value vectors and the training targets are the corresponding coordinate values (2D). For our problem, the RSSI values of WiFi hotpots are formulated into a series of one dimensional vectors, in which each element corresponds to the RSSI value of a WiFi access point. And in the real world, a building may be equipped with a relatively large number of WiFi hotpots to provide good wireless connections, which leads to the problem of high dimensionality. Meanwhile, due to the signal-fading and multi-path effects, WiFi signals are not stable [Hoang et al., 2019]. Therefore, a common neural network based regressor is not powerful enough to describe such complicated relationship between WiFi signals and user location. Moreover, since this task is a sequential prediction, the transition of the hidden states needs to be represented as

well.

The main contributions of our work in this chapter are summarized as follows:

- We devise a novel hybrid deep-learning model which allows us to predict the accurate position of the smartphone users based on detected WiFi fingerprints;
- We conduct the evaluation experiments to compare our method with other deep-learning methods;
- We vary the hyperparameters of the proposed model, i.e., the memory length of the RNN and the mixture number of the MDN, to seek the best performance.

4.2/ METHOD

4.2.1/ CONVOLUTIONAL NEURAL NETWORK

In our first task, the input features are composed of the RSSI values of all the WiFi access points (WAPs) in the buildings, therefore the input dimension can be very high. Since the feature of WiFi fingerprint data represents the different WiFi WAP IDs. The adjacent features suggests that they are close spatially in the real world. Therefore, their RSSI values are more similar when the users are approaching compared to the WAPs that are remote to them (it will be illustrated in the WiFi data samples in the experimental part). For this reason, to deal with the high dimensionality problem, we resort to the technique of Convolutional Neural Networks [LeCun et al., 1998]. CNNs are powerful tools for detecting features and are widely used for tasks such as image processing, natural language processing (NLP) and sensor signal processing.

4.2.1.1/ 1D CONVOLUTIONAL NEURAL NETWORK

In particular, since the input of our model are one dimensional RSSI value vectors, we adopt the 1D Convolutional Neural Network to extract the properties of the high dimensional input. In literature, 1D CNN models are used to process one dimensional data, such as medical data and sensor signal data [Zhao et al., 2019], [Li et al., 2017], [Eren et al., 2019]. In a typical image-processing CNN, the filters are 2 dimensional (if

the input are gray image) or 3 dimensional (if the input are RGB image). While in 1D CNN, the filters are one dimensional, which can be seen as a set of sliding widows. Likewise, the output of the convolutional operations and the max pooling operations will be one dimensional as well.

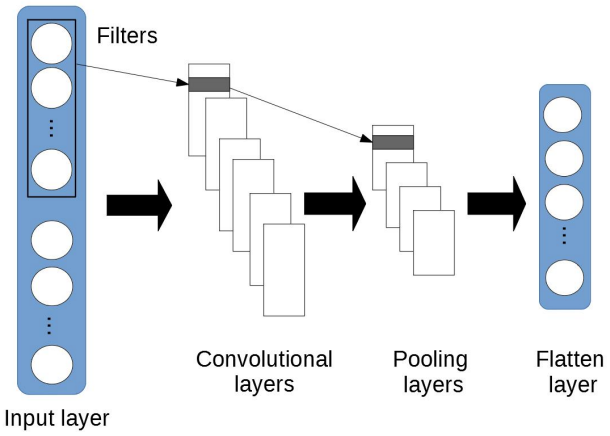


Figure 4.1. The structure of the one dimensional Convolutional Neural Network.

The model structure of one dimensional CNN is illustrated in Fig. 4.1.

4.2.2/ Recurrent Neural Network

To depict the state transitions in the time-series WiFi fingerprint data, we can adopt a deep learning model called Recurrent Neural Network (RNN) [Elman, 1990]. RNNs are widely used for natural language processing (NLP), computer vision and other time series prediction task. In our model, we employ the RNNS to model the complicated relationship between the input (RSSI values) and the output (user location) so as to forecast the user location. As compared to other conventional machine learning model, another advantage of using RNNs is that it is compatible with other deep learning model, such as CNNs. In this section, we briefly review RNNs add their two variants, Long Short-Term Memory networks [Hochreiter et al., 1997] and Gated Recurrent Unit (GRU) [Cho et al., 2014].

4.2.2.1/ VANILLA RNN

The state transitions of RNNs can be expressed as follow

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{4.1}$$

where x_t is the input, h_t is the hidden state, σ_h is the activation function, W_h are the weights from the input layer to the hidden layer, U_h are the weights in the hidden layers and b_h are the biases.

The output of a conventional RNN can be described as follow

$$y_t = \sigma_y(W_y h_t + b_y) \tag{4.2}$$

where y_t is the output, σ_y is the activation function, W_y is the weight and b_y is the output bias.

4.2.2.2/ LSTM

In some situations, RNNs may suffer from the long-term dependency issue during learning process [Hochreiter et al., 1997]. When we try to predict the output at the next time point, we may only need the recent input not any further previous information, in this case a vanilla RNN is capable of handling the problem. But for time-series prediction, in some cases, only the recent information is not enough to learn the tasks, we need to consider the further previous input in order to predict the output at the next time point. This case is called the long-term dependency problem.

To address this issue, researchers proposed a variant of RNNs, called Long Short-Term Memory (LSTM) networks [Hochreiter et al., 1997].The LSTM adopts a special structure which consists of three gates, namely, an input gate, an output gate and a forget gate. These gates regulate the cell states of the LSTM to avoid the long-term dependency problem.

Let C_t be the cell state. The possible value of C_t is between 0 and 1. 1 means that the information is completely kept while 0 means that the information is completely discarded.

Fig. 4.2 depicts the structure of the LSTM.

The first step of LSTM is to compute the output of the forget gate which is to used to decide how much old information will be retained.

$$f_t = \sigma_y(W_y[h_{t-1}, x_t] + b_f) \tag{4.3}$$

where f_t is the output vector of the forget gate, b_f is the bias, [ ] is the concatenation operation.

The second step of the LSTM is to compute the output the input gate. An input layer with a sigmoid activation function, called the input gate layer, is used to decide how much new information will be used.

$$i_t = \sigma_y(W_i[h_{t-1}, x_t] + b_i) \tag{4.4}$$

where i_t is the output vector of the input gate, W_i is the weights and b_i is the bias.

In this step, we update the cell state as well. We need to calculate the candidate value of the cell state, $\hat{C}_t$.

$$\hat{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{4.5}$$

where W_C is the weight and b_C is the bias.

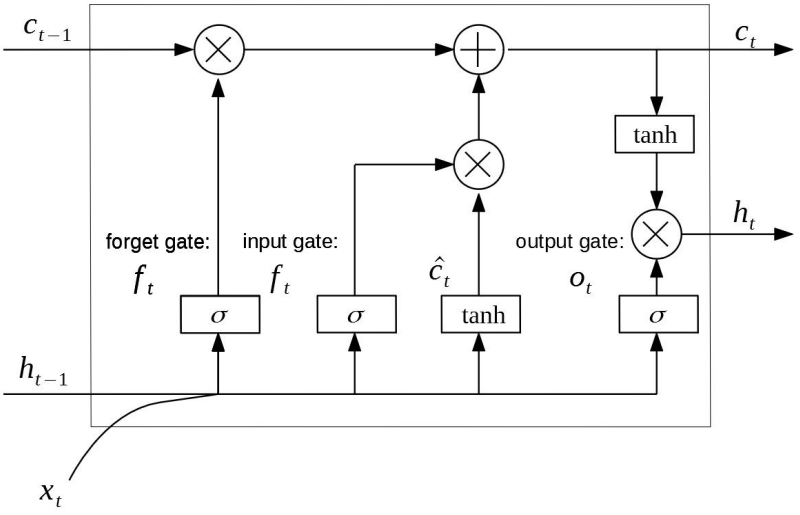


Figure 4.2. The structure of the Long-Short Term Memory network.

In order to update the cell state, we drop the old information based on the output of the forget gate and add the new information base on the output of the input gate.

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{4.6}$$

The last step of the LSTM is to compute the output.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{4.7}$$

where o_t is the activation vector of the output gate, W_o is the weight and b_o is the bias.

Finally, the hidden state is updated as well.

$$h_t = o_t \tanh(C_t) \tag{4.8}$$

4.2.2.3/ GRU

More recently, the researchers proposed a variant of RNNs, Gated Recurrent Units (GRUs) [Cho et al., 2014], [Chung et al., 2014], which has the similar accuracy as LSTMs but less computing cost. The differences between LSTMs and GRUs are that GRUs merge the forget gate and the input gate into one update gate, and merges the cell state and the hidden state into one. By doing so, GRUs have a simpler structure than LSTMs. The learning procedure of the GRU can be summarized as follows:

First, we update the update gate z_t:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{4.9}$$

where W_z is the weight and b_z is the bias.

Then, compute the output r_t:

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{4.10}$$

where W_r is the weight and b_r is the bias.

Afterwards, we compute the candidate hidden state $\hat{h}_t$:

$$\hat{h}_t = \tanh(W_h[h_{t-1}, x_t] + b_C) \tag{4.11}$$

where W_h is the weight and b_C is the bias.

Finally, we update the hidden state:

$$h_t = (1 - z_t) * h_{t-1} + i_t * \hat{h}_t \tag{4.12}$$

where W_h is the weight and b_C is the bias.

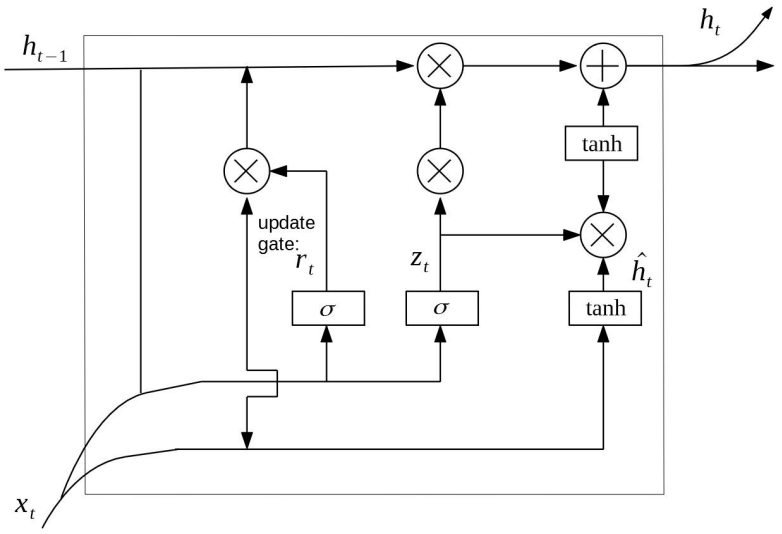


Figure 4.3. The structure of the Gated Recurrent Unit.

Fig. 4.3 depicts the structure of the GRU network.

[Greff et al., 2016] compared a number of the variants of RNNs, and the results show that some variants have better performance than LSTMs on some certain tasks. Therefore, in the latter experiments of this chapter, we will deploy these three RNN architectures in the proposed model respectively for the comparisons.

4.2.3/ Mixture Density Network

A traditional neural network with a loss function, for instance, mean squared errors (MSE), is optimized by a gradient-descent based method. Generally, such model can perform well on the problems that can be described by a deterministic function $f(x)$, i.e., each input corresponds to an output of one specific value. However, for some stochastic problems, one input may map to more than one possible values. Generally, this kind of problems are

better to be described by a conditional distribution $p(y|x)$ than by a deterministic function $y = f(x)$. In such cases, traditional neural networks may not work as expected.

To tackle with this type of problems, we can replace the original loss function with a conditional function. For a regression task, the Gaussian distribution can be a proper choice. Moreover, utilizing the mixed Gaussian distributions can improve the representation capacity of the model. Based on this idea, the researcher proposed Mixture Density Networks (MDNs) [Bishop, Christopher M, 1994]. In contrast with traditional neural networks, the output of a MDN is the parameters of a set of mixed distributions and the loss function becomes the conditional probabilities.

For the target with continuous values (in our case, it is the user coordinates), we can employ a set of Gaussian distributions at the output layer to sample it. Therefore, the optimization process is to minimize the negative log-probability. Hence, the relationship between the input and the output can be described as follow:

$$p(y|x) = \sum_{k=1}^{K} \pi_k p(y|x; \theta_k) \tag{4.13}$$

where x is the input, K is the total mixture number, π_k is the assignment probability for each model, with $\sum_{k=1}^{K} \pi_k = 1, (0 \leq \pi_k \leq 1)$, and θ_k are the internal parameters of the base distribution. For Gaussian distributions, $\theta_k = \{\mu_k, \sigma_k\}$, μ_k are the means and σ_k are the variances.

Accordingly, in the proposed model, the original output layer of the RNN, Eq. (4.2), is rewritten as:

$$\theta_t = \sigma_\theta(W_\theta h_t + b_\theta) \tag{4.14}$$

where θ_t is the output of the RNN sub-model and also the input of the MDN sub-model, σ_θ is the activation function, W_θ are the weights and b_θ are the biases.

After the training process, we can use the neural network along with the mixed Gaussian distributions to represent the target distributions.

4.2.4/ CONVOLUTIONAL MIXTURE DENSITY RECURRENT NEURAL NETWORK

Knowing the merits of the three aforementioned neural networks, we devised a novel deep neural network architecture, called the Convolutional Mixture Density Recurrent Neural Network (CMDRNN). In the CMDRNN model, a 1D CNN is used to capture the features of the high dimensional input, then the state transitions of the time series data is modeled by a RNN model, and the output layer composed of mixed Gaussian densities to enhance the prediction accuracy. With such a structure, we believe that our model is able to illustrate complicated high dimensional time series data. Fig. 4.4 shows the whole structure of the CMDRNN model and Algorithm 2 demonstrates the learning process of the CMDRNN model.

Algorithm 2 Algorithm: CMDRNN

Input: x_t (RSSI Values)

Output: y_t (Coordinates)

while epoch < max epoch **do**
  while i < batch num **do**
    $h_0 \leftarrow Conv1d(x_t)$ ▷ convolutional operation
    $h_1 \leftarrow$ max pool h_0
    $f_t \leftarrow$ flatten h_1
    $h_t \leftarrow \sigma_h(W_h * f_t + U_h * y_{t-1} + b_h)$ ▷ update hidden states
    $\theta_t \leftarrow \sigma_y(W_y * h_t + b_y)$ ▷ compute network output
    $\theta_k \leftarrow \theta_t$ ▷ assign mixture density parameters
    minimize loss function: $-p(y_t|x_t;\theta_k)$
  end while
end while
$y_t \sim p(y_t|x_t;\theta_k)$ ▷ sample final output
return y_t

The uniqueness of our method is that, compared with other existing models in literature, our model adopts a sequential density estimation approach. Thus, the learning target of

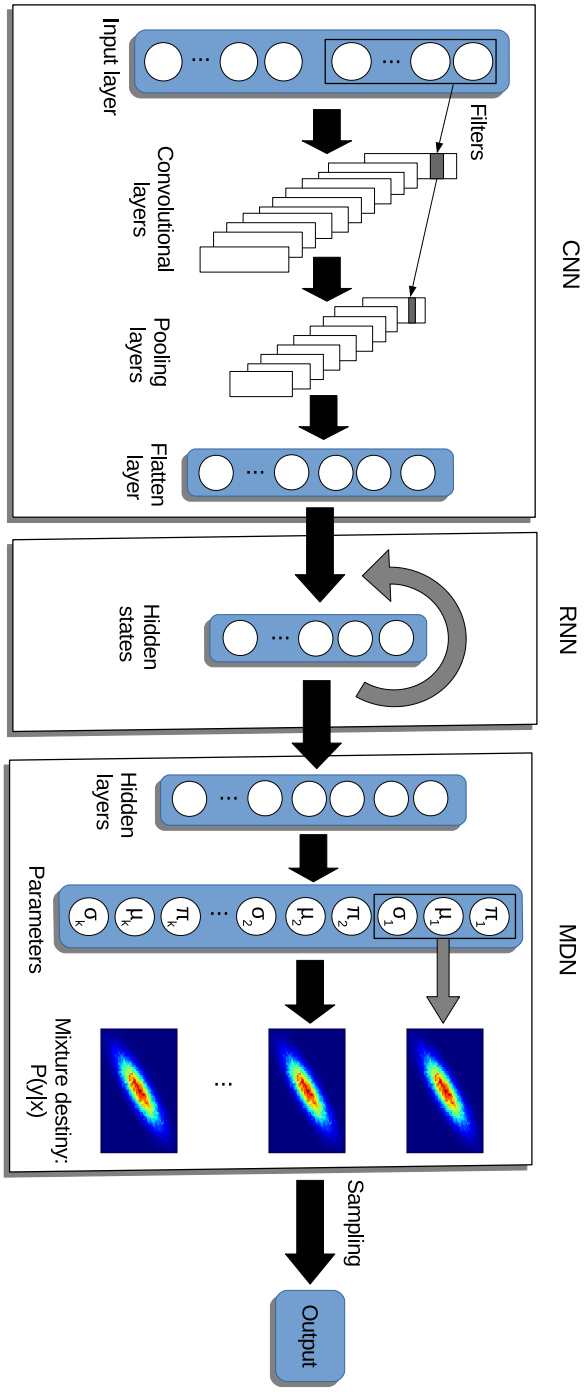


Figure 4.4. The structure of the Convolutional Mixture Density Recurrent Neural Network.

the proposed method becomes a conditional distribution of the data rather than a common regressor. Thanks to this, our model can solve the complicated sequential modeling task in this work.

4.2.5/ OPTIMIZERS

Deep learning models are usually optimized by gradient descent optimization methods [Ruder, 2016]. Here we compare two most widely used optimizers, Adam [Kingma et al., 2015] and RMSProp [Tieleman et al., 2012].

4.2.5.1/ ADAM

Adaptive Moment Estimation (Adam) is a optimization method with adaptive learning rate. Adam can be described as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4.15}$$

where g_t is the gradient and β_1 is the moving the averaging parameter, m_t is the estimation of the mean of the gradients.

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4.16}$$

where β_2 is the moving the averaging parameter, v_t is the estimation of the variance of the gradients.

The bias of the m_t is alleviated by

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.17}$$

The bias of the v_t is alleviated by

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4.18}$$

Finally, the parameters θ are updated by

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{4.19}$$

where η is the learning rate.

4.2.5.2/ RMSProp

RMSProp keeps the moving average of the squared gradients for each weight and it divides the gradient of the root mean square (RMS). RMSProp can be described as follows:

$$\mathbb{E}[g^2]_t = \beta\,\mathbb{E}[g^2]_{t-1} + (1-\beta)\left\{\frac{\delta C}{\delta w}\right\}^2 \tag{4.20}$$

where $\mathbb{E}[g^2]$ is the moving average of the square gradient, β is the moving average parameter, usually set to be 0.9, $\frac{\delta C}{\delta w}$ is the gradient the loss function with respect to the weight.

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{\mathbb{E}[g^2]_t}}\frac{\delta C}{\delta w} \tag{4.21}$$

where η is the learning rate.

In the experimental parts, we will adopt both Adam and RMSProp as the optimizers in order to see which performs better on the learning tasks.

4.3/ Experiments and Results

4.3.1/ Dataset Description

For the validation dataset, we select two WiFi RSSI-Coordinate paths from the Tampere dataset [Lohan et al., 2017a]. As shown in Fig. 4.5 The input dimension of the Tampere dataset is 489. The RSSI values of the detected WAPs range from -110 dB to 0 dB and the RSSI values of undetected WAPs are set to be 100. The units of the target values are meters. For pre-processing the data, we set the undetected values to 0.

Input Output

Timestamp	WAP001	WAP002	···	WAP076	WAP077	···	WAP489	X	Y
0	100	100	···	-98	-89	···	100	179.43	71.716
1	100	100	···	-93	-87	···	100	180.43	72.716
2	100	100	···	-91	-76	···	100	181.43	72.716
⋮				⋮				⋮	
N	-77	-75	···	100	100	···	100	185.43	74.716

Figure 4.5. WiFi fingerprint data samples.

4.3.2/ Model Implementation Details

The implementation details of our model are illustrated in Table 4.1. The CNN sub-network consists of three layers, a convolutional layer, a max pooling layer and a flatten layer. The RNN sub-network includes a hidden layer with 200 neurons. The MDN sub-network has a hidden layer and output layer. The mixed Gaussians number of the MDN output layer is 30, and each mixture has 5 parameters, namely, two dimensional means, diagonal variances and corresponding weights. For the optimizer, we choose RMSProp.

4.3.3/ Choice of Hypermeters

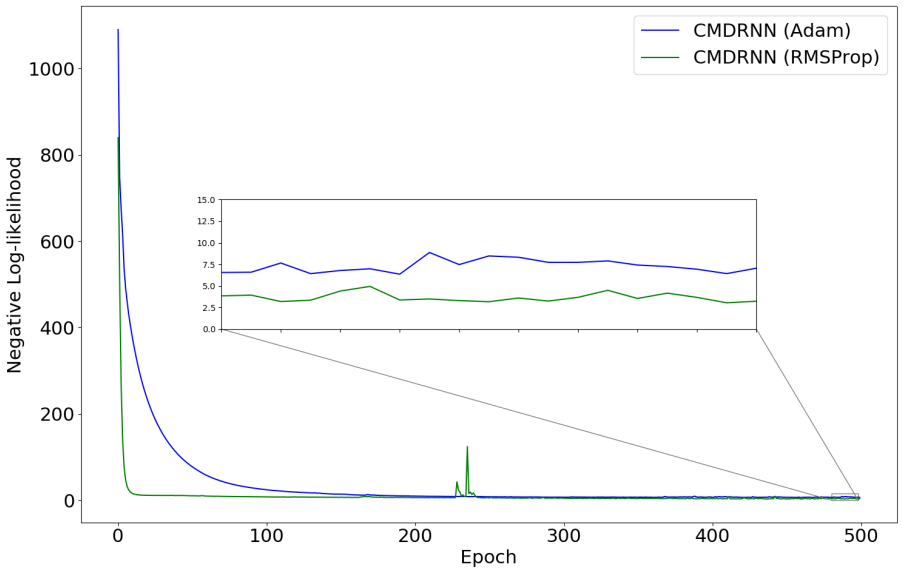


Figure 4.6. Training losses using different optimizers.

Table 4.1. CMDRNN Implementation Details

Sub-network	Layer	Hyperparameter	Activation Function
CNN	Convolutional layer	filter number: 100; stride: 2	Sigmoid
CNN	max pooling layer		ReLU
CNN	flatten layer	neuron number: 100	Sigmoid
RNN	hidden layer	neuron number: 100	Sigmoid
MDN	hidden layer	memory length: 5; neuron number: 200	Leaky ReLU
MDN	output layer	neuron number: 200	-
		5*mixed Gaussian number (5*30)	
Optimizer: RMSProp; learning rate: 10^{-3}			

In [Martin Arjovsky et al., 2017], it reports that RMSProp [Tieleman et al., 2012] may have better performance on very non-stationary tasks than the Adam optimizer [Kingma et al., 2015]. To verify this, we train our algorithm with RMSProp and Adam, respectively. As it is shown in Fig. 4.6, the proposed model can converge to a lower negative log-likelihood via RMSProp than Adam. Thus, we choose RMSProp as the optimizer for our model.

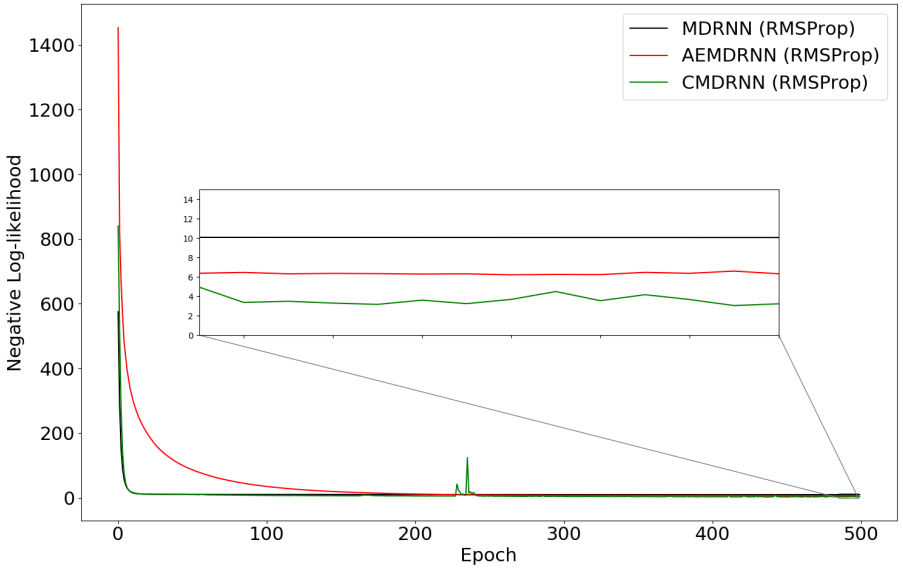


Figure 4.7. Training losses using different feature detectors.

Fig. 4.7 exhibits different results of using different feature detectors.

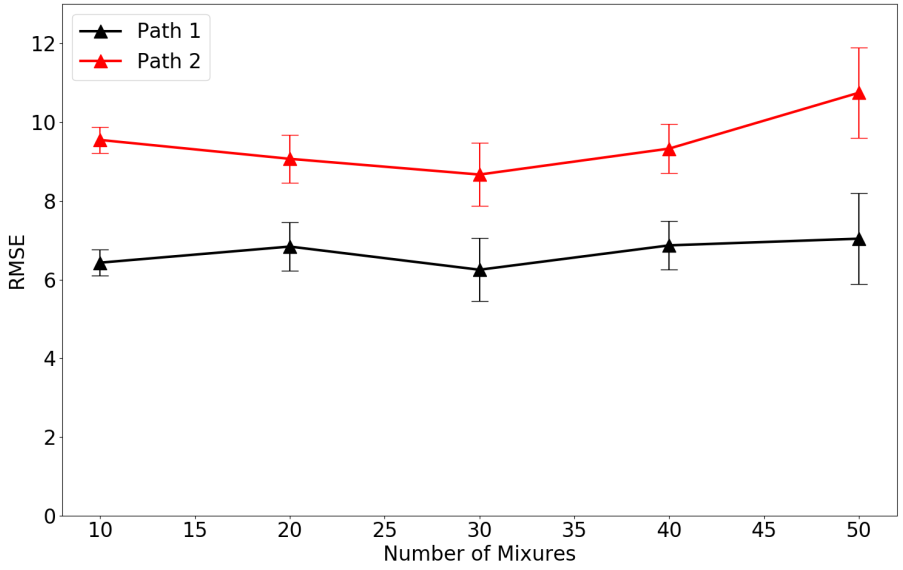


Figure 4.8. Prediction results of different mixture numbers in the MDN (bars represent the standard deviations).

Fig. 4.8 exhibits different results of using different mixture numbers at the output layer of the MDN. We can see that the most appropriate number is 30.

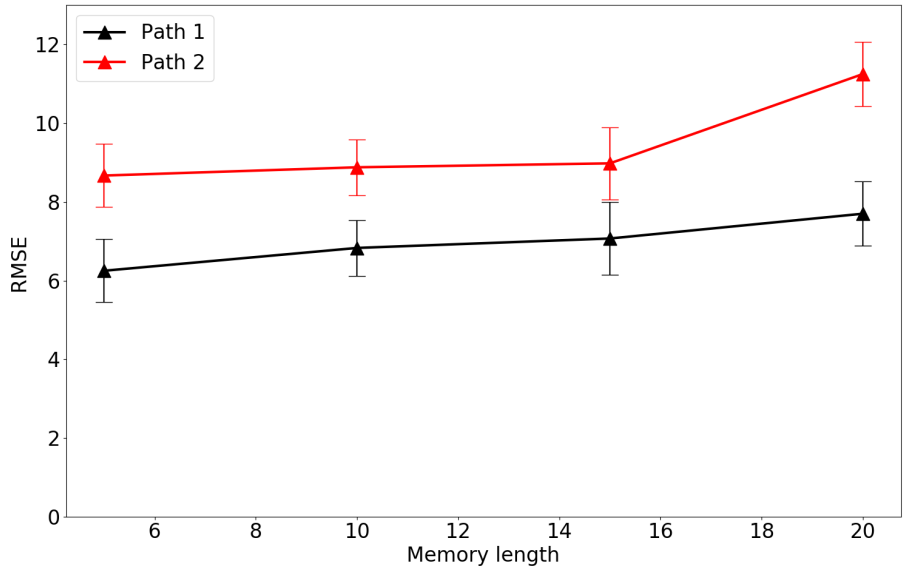


Figure 4.9. Prediction results of different memory lengths in the RNN (bars represent the standard deviations).

Fig. 4.9 demonstrates different results of using different memory lengths at the output layer of the MDN. We can see that when the memory length is shorter, the result is better, thus we set the memory length of our model to be 5.

4.3.4/ Comparisons with Other Methods

In order to evaluate the effectiveness of our method, we conducted several experiments to thoroughly compare our CMDRNN model to other deep learning approaches. K-NN, DT and RF are used as the baseline models [Rojo et al., 2019]. The purposes of experiments are indicated as follows:

- Comparing different optimizers: Adam v.s. RMSProp;
- Comparing different feature detectors: RNN, RNN+MDN, AE + RNN + MDN, CMDRNN;
- Comparing different regressors: RNN, CNN+RNN and CMDRNN;
- Comparing RNN variants: CMDRNN, CMDLSTM and CMDGRU.

Fig. 4.10 and Fig. 4.11 shows the predicting results of two selected paths, respectively.

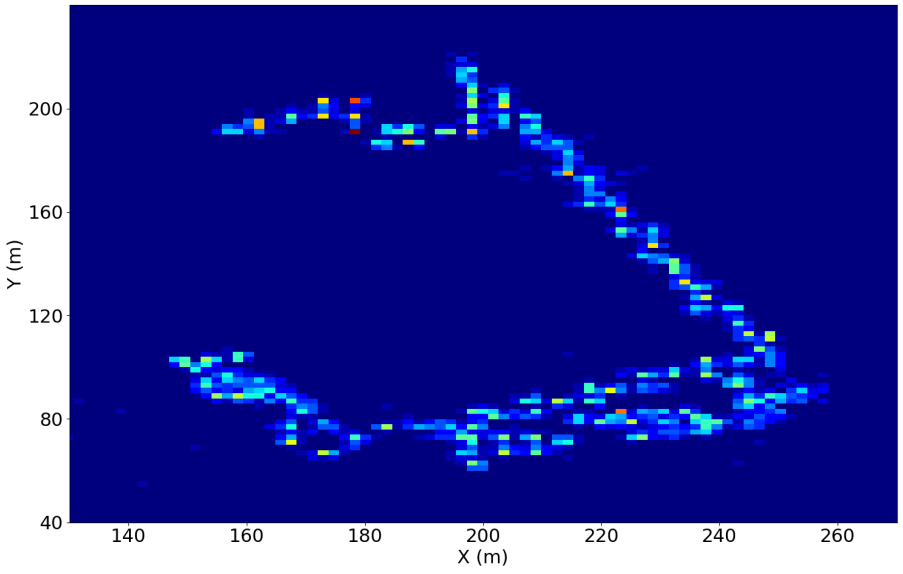


Figure 4.10. Path 1 prediction results.

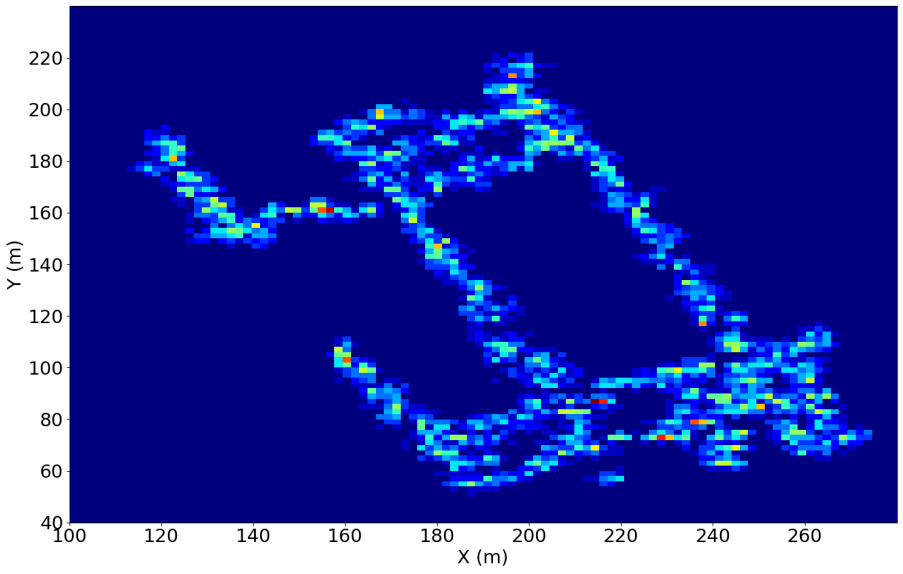


Figure 4.11. Path 2 prediction results.

Table 4.2. Root mean squared errors of the path prediction results (meter).

(a) Results of the baseline models.

Path	k-NN	DT	RF
Path 1	7.44 ± 0.00	8.78 ± 0.76	7.25 ± 0.25
Path 2	8.02 ± 0.00	20.94 ± 1.52	9.60 ± 0.75

(b) Results of the sequential prediction models.

Path	RNN	CNN+RNN	RNN+MDN	AE+RNN+MDN
Path 1	29.36 ± 1.61	34.26 ± 3.04	23.86 ± 5.50	11.24 ± 0.86
Path 2	31.61 ± 0.74	36.75 ± 6.17	23.58 ± 2.29	12.01 ± 1.68

(c) Results of the proposed models.

Path	CMDRNN	CMDLSTM	CMDGRU
Path 1	8.26 ± 1.31	7.38 ± 0.89	6.25 ± 0.80
Path 2	10.17 ± 0.72	9.26 ± 0.31	8.67 ± 0.23

The overall results are demonstrated in Table 4.2.

Since the input is high dimensional, the sagacious way to deal with this is to incorporate a sub-network into the model for dimension reduction or feature detection. Many previous research adopted auto-encoders to reduce dimension, while we argue that the more appropriate choice for the task in our work is using a one-dimensional CNN. In order to prove that, we test three different models, one without a feature-detecting structure, one using an Autoencoder and one using 1D CNN (the proposed model). The auto-encoder model with structure $\{$hidden neurons: 256; hidden neurons: 128; code size: 64; hidden neurons: 128; hidden neurons: $256\}$.

In the experiments, we used three baseline models, k-NN, Decision Tree and Random Forests (which are not sequential models). We run the algorithms multiple times with random initialisation. From Table 4.2, we can see that, compared to the baseline models, our proposed models (which are sequential models) have comparable performances, especially for CMDGRU.

In addition, for sequential predicting models, as the results illustrated Fig. 4.7 and Table 4.2, the proposed model with 1D CNN feature detector can reach lower negative log-likelihood during the training process and has the smallest RMSE on the test data,

respectively.

4.4/ CONCLUSION

In this chapter, we attempt to tackle with the WiFi fingerprint-based user positioning problem. In contrast with existing approaches, our solution is a hybrid deep learning model. The proposed model is composed of three deep neural network, a CNN, a RNN and a MDN. This unique deep architecture combines all the strengths of three deep learning models, which enables us to recognize and predict user location with high accuracy. Finally, we tested our model on the real-world dataset and found the optimal hyperparameters for the CMDRNN models. The obtained results verifies the effectiveness of our approach and shows the superiority of our methods compared other deep learning-based methods as well.

5

Recognizing Indoor Location via Semi-supervised Learning

5.1/ Introduction

In the previous chapter, we have studied how to use WiFi fingerprint data to predict user location at next time points. We employed a hybrid deep learning model to tackle with the problem. Although the obtained results are satisfying, we still want to improve the accuracy of the location recognition even further by exploring other advanced machine learning techniques.

In this chapter, our research goal is to calculate more accurate user location via the relevant WiFi fingerprints. That is to say, we use the WiFi RSSI value data as the input and the actual user location (latitudes and longitudes) as the output. This problem can be regarded as a high dimensional regression problem without considering the temporal order of the data.

However, in order to achieve this goal, there are some difficulties, such as the signal-fading and multi-path effects, as we mentioned in the previous chapter. Therefore, if we use a Conventional Neural Network model to solve this problem directly, the existence of the noisy information within the input data is detrimental to the modeling accuracy. This prevents us from computing the user location more precisely.

Hence, we need to utilize a specific method to extract the task-related information from the input. Previous deep learning methods, such as Autoencoders and Convolutional Neural Networks [Nowicki et al., 2017], [Ibrahim et al., 2018] are not powerful enough to accom-

plish this task. To circumvent this problem, we suggest that learning a representation of the input via unsupervised learning first can extract the useful task-related information effectively. Based on this idea, we propose a novel semi-supervised deep learning method for accurate indoor user location recognition.

The main contributions in this chapter are summarized as follows:

- We propose a Variational Autoencoder-based semi-supervised learning model;
- We conduct a series of experiments with different amounts of labeled data to evaluate the proposed model;
- We compare our model with other existing machine learning and deep learning models.

5.2/ METHOD

To accomplish the task, we presume that the input (WiFi fingerprints) and the output (GPS coordinates) are related to the same underlying variable related to the physical user location. Additionally, according to the Bayesian Central Limit Theorem, the posterior probability distribution is approximately a normal distribution under certain circumstances. Based on this, we can leverage a Deep Latent Generative Model (DLGM) to learn the latent distribution with the input data in an unsupervised manner. In fact, we find that a Variational Autoencoder (VAE) model [Kingma et al., 2013] can be a very good choice for this task.

Once we have the learned latent distribution of the input, we utilize it as the input to feed a predictor model. The predictor can simply be a regression neural network. This learning procedure is supervised. Combining the unsupervised learning and supervised procedures, we now devise a semi-supervised learning method. The advantage of using a semi-supervised learning model is that we can make use of not only the labeled data but also the unlabeled data to improve the location recognition accuracy. This learning scheme can be very useful when we have a relatively large amount of unlabeled data but a relatively small amount of labeled data, which occurs in many real-world cases.

5.2.1/ Model Setup

From a probabilistic perspective, in order to let the proposed semi-supervised learning method work, we need to make following assumptions first:

- **Assumption** 1: There a statistic $t(x)$ solely of x that is enough to be the sufficient statistic for z, i.e., $t(x)$ captures all the necessary information for calculating the parameters in the distribution of z.

- **Assumption** 2: There exists a statistic $t(z)$ solely of z that is enough to be the sufficient statistic for y, i.e, $t(z)$ captures all the necessary information for calculating the parameters in the distribution of y.

- **Assumption** 3: We assume the marginal distribution $q(z)$ is a normal distribution (because Bayesian Central Limit Theorem says that under certain circumstances, the posterior probability distribution is approximately a normal distribution).

Assumption 1 describes the relationship between the input x and the latent variable z and explain why we can infer the distribution of the latent variable z with the input x. This is the theoretical fundamental for the unsupervised learning process in our model. Similarly, the second assumption indicates why we can use the latent variable z to calculate the target y. This is the theoretical fundamental for the supervised learning process in our model.

To understand our assumptions better, here we use a example to briefly explain what a sufficient statistics is. For a Gaussian distribution with known variance and unknown mean μ, if we want to estimate μ, we can use the sample mean as the estimate of this mean. Then the sample mean is the a sufficient statistics of μ.

In addition, it is a practical reason for doing so. Since in many real-world cases, the available datasets have more information about the input x and less information about the target y, therefore it is more reliable to infer the latent distribution of z, $q(z)$, via $p(z|x)$ rather than via $p(z|y)$.

According to the chain rule and with the assumptions we made, then the generative

scheme can be formulated as follow:

$$\begin{aligned} p(y,z,x) &= p(y|z,x)p(z,x) \\ &= p(y|z,x)p(z|x)p(x) \end{aligned} \tag{5.1}$$

From the perspective of Monte Carlo sampling, in Eq. (5.1), $p(x)$ can be approximated by drawing the input samples from the dataset D, $p(z|x)$ can be a neural network-based encoder. Accordingly, the predictor model can be described as:

$$y \sim p(y|z,x) \tag{5.2}$$

Eq. (5.2) can be learned by a multi-layer perceptron (MLP), which can be either deterministic or a probabilistic in practice.

To construct the proposed model, we can implement Eq. (5.1) and Eq. (5.2) through combining an unsupervised learning process and a supervised learning process. Hence, our method consists of two learning steps:

- The first step (unsupervised learning): we employ a deep generative model, e.g., a VAE model, to learn the latent distribution $p(z|x)$;
- The second step (supervised learning): we employ a MLP predictor and use the learnt latent variable to learn the target y.

5.2.2/ UNSUPERVISED LEARNING PROCEDURE

For the unsupervised learning, we adopt a Variational Autoencoder as the generative model used to learn the latent distribution. Variational Autoencoders (VAEs) [Doersch, 2016], [Kingma et al., 2014b] are deep latent generative models which adopt Variational Inference. Different to conventional autoencoders or other generative models, the latent representations of VAEs are continuous probabilistic distributions, which can be used to represent the real user coordinates.

In this part, we introduce the background of VAEs briefly. VAEs are similar to conventional Autoencoder (AEs) in a sense, it also is a unsupervised learning model which consists of a encoder neural network and an decoder neural network. The difference between AEs

and VAEs is that the latent codes of the AE are deterministic, thus the AE usually can be used to reduce the input dimension but not to generate new data samples.

In contrast, the latent codes of VAEs are stochastic thus VAEs can be used to generate new data samples. Additionally, more complex posteriors of VAEs can be constructed by a kind of methods called Normalizing Flows (NFs) [Rezende et al., 2015], through bijective mappings, two simple flows, the planar flow and radial flow. Or to compute even more complicated flows, we can resort to the autoregressive flow methods, such as, the Masked Autoregressive Flows (MAFs) [Papamakarios et al., 2017] and Inverse Autoregressive Flows (IAFs) [Kingma et al., 2016].

5.2.2.1/ Evidence Lower Bound of VAEs

VAEs are originally proposed to sample new data samples. But direct computing the evidence of data, $p(x)$, is difficult, we need to adopt Variational Inference to approximate it. To this end, we need to derive of the evidence lower bound (ELBO). The derivation is as follows:

$$\begin{aligned}
\log p(x) &= \mathbb{E}_{q(z|x)}[\log p(x)] \\
&= \mathbb{E}_{q(z|x)}\left[\log\left\{\frac{p(x,z)}{p(z|x)}\right\}\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log\left\{\frac{p(x,z)q(z|x)}{q(z|x)p(z|x)}\right\}\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log\left\{\frac{p(x,z)}{q(z|x)}\right\}\right] + \mathbb{E}_{q(z|x)}\left[\log\left\{\frac{q(z|x)}{p(z|x)}\right\}\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log\left\{\frac{p(x,z)}{q(z|x)}\right\}\right] + D_{KL}(q(z|x)||p(z|x))
\end{aligned} \tag{5.3}$$

Since the Kullback-Leibler divergence $D_{KL}(q(z|x)||p(z|x)) \geq 0$, then we have

$$\begin{aligned}
\log p(x) &\geq \mathbb{E}_{q(z|x)}\left[\log\left\{\frac{p(x,z)}{q(z|x)}\right\}\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log\left\{\frac{p(x|z)q(z)}{p(z|x)}\right\}\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log p(x|z)\right] + \mathbb{E}_{p(z|x)}\left[\log\left\{\frac{q(z)}{q(z|x)}\right\}\right] \\
&= \mathbb{E}_{q(z|x)}\left[\log p(x|z)\right] - D_{KL}(q(z|x)||q(z))
\end{aligned} \tag{5.4}$$

where $q(z)$ is the prior of the latent variable z, which can be regarded as a standard Gaussian distribution

$$z \sim \mathcal{N}(0, \mathbb{I}) \tag{5.5}$$

Now we have the evidence lower bound as the optimization objective for VAEs.

5.2.2.2/ Learning Method of VAEs

In order to build a VAE model, we can use an encoder parameterized by ϕ, $p_\phi(z|x)$, to represent the posterior, and a decoder parameterized by θ, $p_\theta(z|x)$, to represent the reconstructing likelihood. Note that Eq. (5.4) cannot be computed explicitly, but we can leverage the reparameterization trick [Kingma et al., 2014b] and Monte Carlo sampling to solve it.

According the VAE decoder used to compress the input can be described as:

$$z \sim p_\phi(z|x) \tag{5.6}$$

In order to use a neural network to learn the posterior $p_\phi(z|x)$, the encoder needs to be differentiable. To this end, the posterior sample z is reparameterized by using the following equation:

$$z = \mu_z + \sigma_z \odot \epsilon, \ \epsilon \sim \mathcal{N}(0; \mathbb{I}) \tag{5.7}$$

where μ_z is the mean of z, σ_z is the variance of z, $\odot$ is the Hadamard product, ϵ is the noise.

The VAE decoder used to reconstruct the original input can be described as:

$$x' \sim p_\theta(x|z) \tag{5.8}$$

where x' is the reconstructed input.

Since maximizing ELBO is equivalent to minimizing the loss function of the VAE, then accordingly, the loss function of the VAE yields:

$$\mathcal{L}(x, \theta, \phi) = \mathbb{E}_{z \sim p_\phi(z|x)} \left[-\log(p_\theta(x|z)) \right] + D_{KL}(p_\phi(z|x) \| q(z)) \tag{5.9}$$

Once $\mathcal{L}(\theta, \phi, D)$ is minimized, we can have the approximate posterior $p_\phi(z|x)$ for sampling the latent variable z. Note that in the unsupervised learning process, we can use all the input data whether it is labeled or not to train the VAE model. In this way, we take advantage of the available data maximally.

5.2.3/ Supervised Learning Procedure

After the unsupervised learning procedure, we can compute the latent distribution of z via the encoder. In the following step, we utilize the WiFi RSSI data as the input x and the user coordinates as the target y to proceed the supervised learning procedure. To this end, we devise two predicting models, one is a deterministic model and the other is a stochastic model.

5.2.3.1/ Deterministic Predictor (M1 Model)

In the first model, we build a deterministic predictor which consists of two steps:

Step 1: to obtain the mean of the Gaussian distribution of latent variables.

$$\mu_z = \mathcal{F}_\mu(x; \phi) \tag{5.10}$$

where $\mathcal{F}_\mu(x; \phi)$ can be regarded as the encoder of the VAE.

Step 2: to obtain the final prediction based on the output of Step 1.

$$y = \mathcal{F}_y(\mu_z; w) \tag{5.11}$$

where $\mathcal{F}_y(\mu; w)$ is a deterministic multi-layer perceptron model. Consequently, the loss function is

$$\mathcal{L}(D; w) = \frac{1}{N} \sum_{n=1}^{N} (\hat{y}_n - y_n)^2 \tag{5.12}$$

where $\hat{y}_n$ is the labeled target.

Compared to using the original data as the input to compute the target directly, using the learnt latent distribution as the input of the predictor can reduce the noisy information of the original input. As a result, our model can alleviate the overfitting problem. The loss

function of the M1 model is the mean squared errors. The training scheme of the M1 model is summarized in Algorithm 3.

Algorithm 3 Algorithm: M1 model

Input: x_a (all input), x_l (labeled input), $\hat{y}$ (labels)

Output: y (predictions)

while unsupervised learning **do**
  $\mu_z, \sigma_z \leftarrow E_\phi(x_a)$ ▷ $E_\phi(x_a)$: Encoder networks
  $z \sim \mathcal{N}(\mu_z, \sigma_z)$ ▷ Sample latent codes
  $x'_a \leftarrow p(x|z;\theta)$ ▷ x'_a: reconstructed input
  minimize loss function $\mathcal{L}(\theta, \phi, D)$ ▷ Eq. (5.9)
end while

while supervised learning **do**
  $\mu_z \leftarrow \mathcal{F}_\mu(x_l;\phi)$ ▷ get latent codes
  $y \leftarrow \mathcal{F}_y(\mu_z; w)$ ▷ get predictions
  minimize loss function $\mathcal{L}(D; w)$ ▷ Eq. (5.12)
end while
return y

5.2.3.2/ PROBABILISTIC PREDICTOR (M2 MODEL)

Alternatively, in contrast with the M1 model, we can also devise a stochastic predictor, the M2 model whose loss function is the negative likelihood as opposed to the M1 model. To this end, based on Eq. (5.1), we factorize the joint distribution:

$$p(y, z, x; w, \phi) = p_w(y|z, x)p_\phi(z|x)p(x) \tag{5.13}$$

where $p_\phi(z|x)$ is the encoder network parameterized by ϕ, and $p_w(y|z, x)$ the predictor network parameterized by w, $p(x)$ can be approximated via empirically drawing the samples from the dataset.

Based on Eq. (5.13), we can formulate a probabilistic prediction model. However, since Eq. (5.13) cannot be computed explicitly, we can use Monte Carlo method to solve to it

by drawing the samples of z and y. To this end, first, we draw the latent variables z from the VAE encoder according to Eq. (5.6). Then, we draw the predicted values y by using the conditional probability:

$$y \sim p_w(y|z, x) \tag{5.14}$$

Hence based on Eq. (5.9) and Eq. (5.13), the total loss function of the M2 model can be written as:

$$\mathcal{L}(D; \theta, \phi, w) = \mathbb{E}_{z\sim p_\phi(z|x)}[-\log p_w(y|z, x)] + \mathbb{E}_{z\sim p_\phi(z|x)}[-\log p_\theta(x|z)] + D_{KL}(p_\phi(z|x)||q(z)) \tag{5.15}$$

In Eq. (5.15), the first term represents the predictor, the second term represents the decoder and the last term represents the encoder. The second term and the last term can be optimized by the unsupervised procedure at the first step.

Since ϕ and θ are trained, according to Eq. (5.15), here we only need to optimize $p_w(y|z, x)$ at the second step. Thus, the loss function for training the predictor becomes:

$$\mathcal{L}(D; w) = \mathbb{E}_{z\sim p_\theta(z|x)}[-\log p_w(y|z, x)] \tag{5.16}$$

In practice, we can use Monte Carlo sampling to solve the above loss:

$$\mathcal{L}(D; w) \approx -\frac{1}{N}\sum_{n=1}^{N} \log p_w(y|z, x) \tag{5.17}$$

where N is the mini batch size.

Furthermore, in contrast with the deterministic M1 model, in order to build a stochastic predictor, we assume that the likelihood function $p_w(y|z)$ is a Gaussian distribution with noise σ_y which can be seen as a hyper-parameter. For predicting, we draw multiple samples via the predictor and use their mean value as the final output.

Algorithm 4 Algorithm: M2 Model

Input: x_a (all input),x_l (labeled input),y_l (labels)
Output: y (predictions)

while unsupervised learning **do**
  $\mu_z, \sigma_z \leftarrow E_\phi(x_a)$ ▷ $E_\phi(x_a)$: Encoder networks
  $z \sim \mathcal{N}(\mu_z, \sigma_z)$ ▷ Sample latent codes
  $x'_a \leftarrow p_\theta(x|z)$ ▷ x'_a: reconstructed input
  minimize loss function $\mathcal{L}(\theta, \phi, D)$ ▷ Eq. (5.9)
end while

while supervised learning **do**
  $z \sim \mathcal{N}(\mu_z, \sigma_z)$ ▷ Sample latent codes
  $y \sim p_y(z; w)$ ▷ Sample predictions
  minimize loss function $\mathcal{L}(D; w)$ ▷ Eq. (5.17)
end while
return y

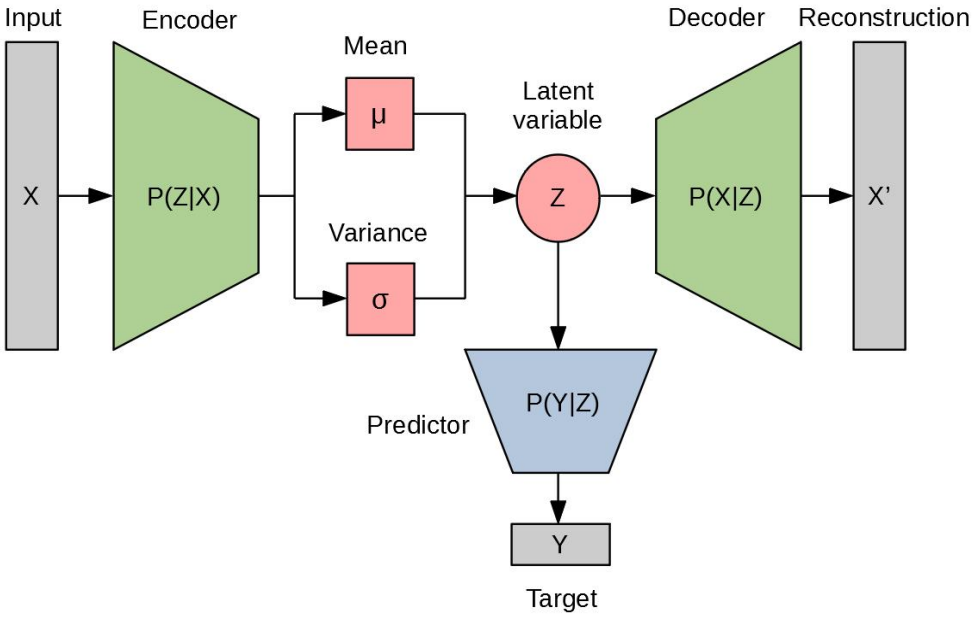


Figure 5.1. VAE-based semi-supervised learning model.

The training scheme of the M2 model is summarized in Algorithm 4 and the structure of the VAE-based semi-supervised learning model is illustrated in Fig. 5.1.

5.3/ Experiments and Results

5.3.1/ Dataset Description

For the validation dataset, we use the UJIIndoorLoc dataset [Torres-Sospedra et al., 2014]. The input dimension of the UJIIndoorLoc dataset is 520. The RSSI values of the detected WAPs range from -100 dB to 0 dB and the RSSI values of undetected WAPs are set to be 100. The coordinates are in longitudes and latitudes. The total instances number for the experiments is about 20000.

For pre-processing the data, we set the undetected values into 0 and remove the duplicate instances. The original target data are longitudes and latitudes with very large values. They are scaled for the experiments, thus the predicting results in Table 5.2 do not have units. We run the algorithms multiple times with random initialisation.

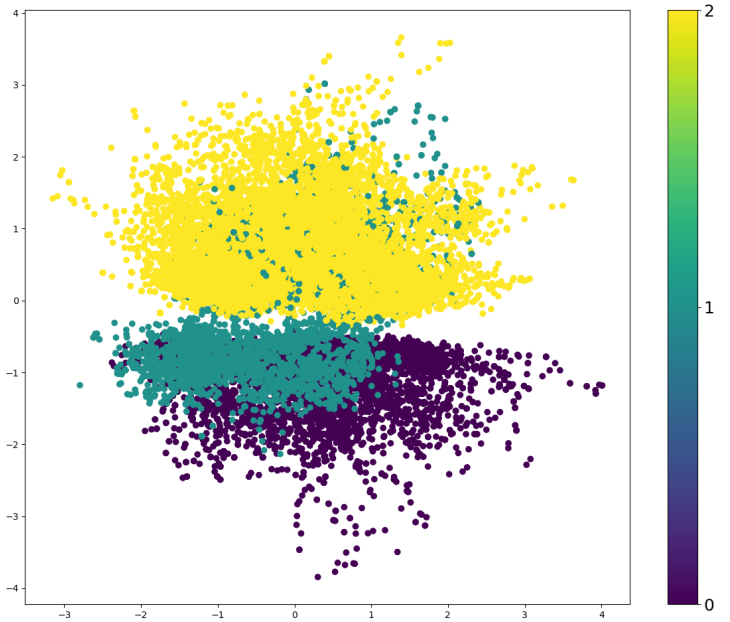
5.3.2/ Model Implementation Details

The VAE-based model consists of three sub-networks, the encoder, the decoder and the predictor. The implementation details of the VAE-based semi-supervised learning model are demonstrated in Table 5.1.

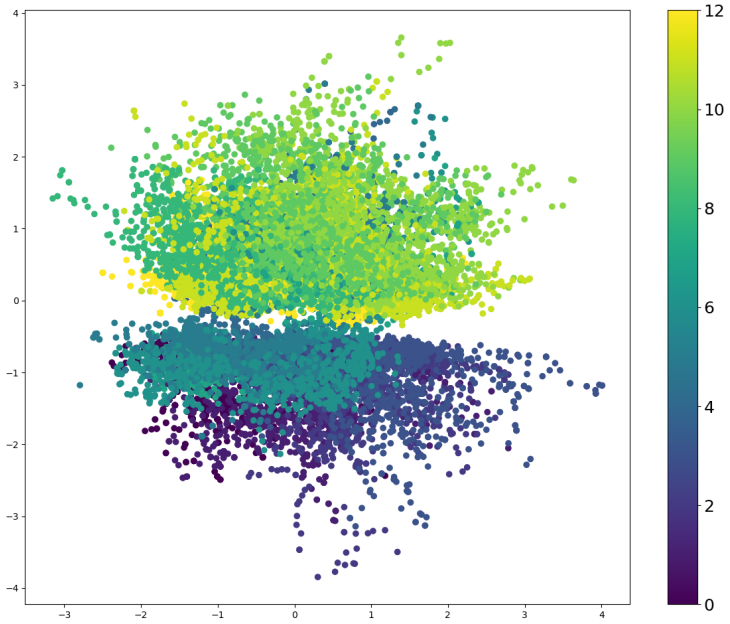
Table 5.1. VAE-based model implementation details

Sub-network	Layer	Hyperparameter	Activation Function
Encoder	hidden layer	neuron number: 512;	ReLU
Encoder	hidden layer	neuron number: 512; latent dimension: 5	ReLU
Decoder	hidden layer	neuron number: 512	ReLU
Predictor	hidden layer	neuron number: 512; dropout rate: 0.3	ReLU
Predictor	hidden layer	neuron number: 512; dropout rate: 0.3	ReLU
Predictor	hidden layer	neuron number: 512; dropout rate: 0.3	ReLU
Optimizer: Adam; learning rate: 10^{-3}			

5.3.3/ RESULTS



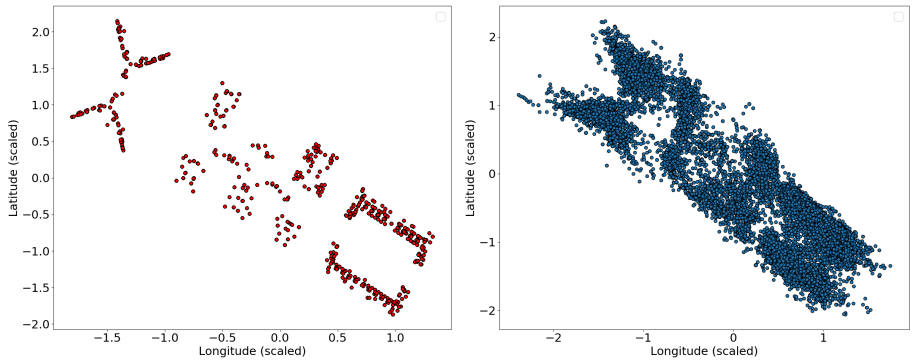
(a) Latent variables labeled with the building IDs, here shows the 2D projection.



(b) Latent variables labeled with the floor IDs, here shows the 2D projection.

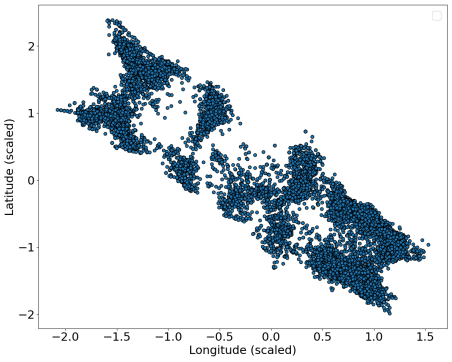
Figure 5.2. Latent variables with dimension of 5, here shows the 2D projection.

Fig. 5.2 demonstrate the distribution of latent variable z. We can see that the latent distribution is related to the building IDs and the floor IDs.

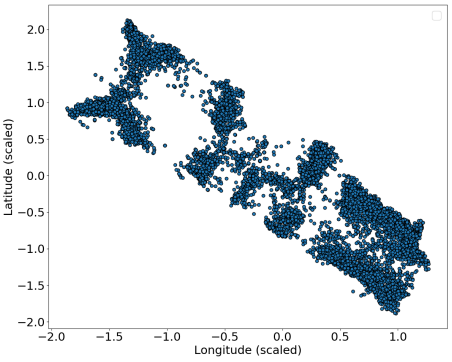


(a) Ground truth.

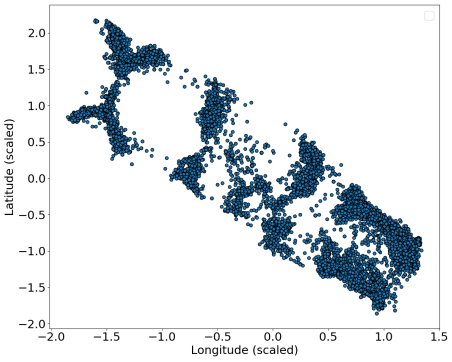
(b) Labeled data: 2%.



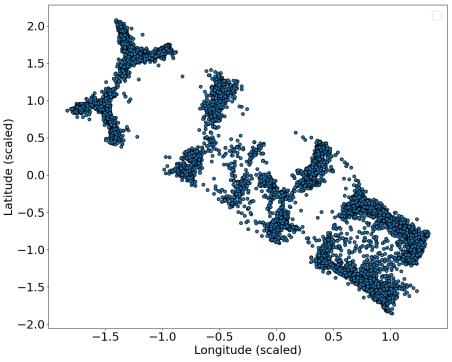
(c) Labeled data: 5%.



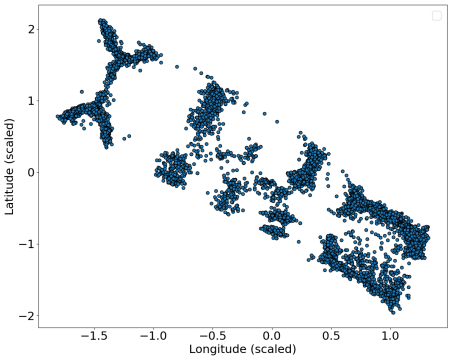
(d) Labeled data: 10%.



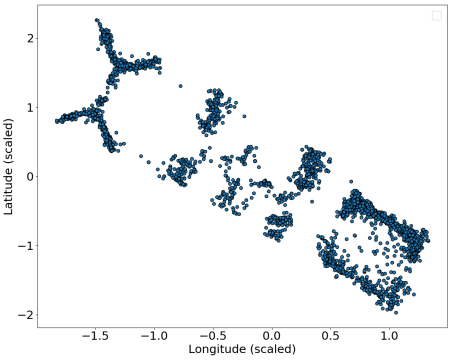
(e) Labeled data: 20%.



(f) Labeled data: 30%.



(g) Labeled data: 50%.



(h) Labeled data: 80%.

Figure 5.3. Testing results for M2 model.

For the experimental set up, we use different portions of the labeled data, ranging from

Table 5.2. Root mean squared errors of testing results with different portions of labeled data

Labeled data	k-NN	DT	RF	GP	MDN(2)	MDN(5)	M1	M2
2%	0.202 ± 3e-2	0.349 ± 5e-2	0.234 ± 3e-2	0.565 ± 3e-2	0.161 ± 5e-3	0.159 ± 4e-3	0.175 ± 6e-3	0.166 ± 8e-3
5%	0.177 ± 1e-2	0.254 ± 2e-2	0.184 ± 2e-2	0.313 ± 2e-2	0.139 ± 4e-3	0.143 ± 3e-3	0.133 ± 3e-3	0.123 ± 4e-3
10%	0.156 ± 6e-3	0.201 ± 2e-2	0.138 ± 6e-3	0.275 ± 3e-3	0.120 ± 6e-3	0.129 ± 1e-3	0.105 ± 2e-4	0.106 ± 5e-3
20%	0.120 ± 4e-3	0.161 ± 9e-3	0.112 ± 5e-4	0.262 ± 8e-4	0.107 ± 6e-3	0.105 ± 4e-3	0.093 ± 2e-3	0.093 ± 2e-3
30%	0.104 ± 2e-3	0.140 ± 4e-2	0.100 ± 2e-3	0.258 ± 2e-3	0.102 ± 5e-3	0.105 ± 4e-3	0.086 ± 3e-3	0.087 ± 2e-3
50%	0.100 ± 1e-2	0.126 ± 2e-3	0.091 ± 2e-3	0.253 ± 2e-3	0.101 ± 7e-3	0.097 ± 4e-3	0.080 ± 9e-4	0.083 ± 4e-3
80%	0.092 ± 7e-3	0.112 ± 3e-3	0.087 ± 3e-3	0.253 ± 2e-3	0.098 ± 7e-3	0.103 ± 3e-3	0.077 ± 4e-3	0.079 ± 3e-3

2% to 80%. We use k-NN, DT and RF as the baseline models and use GP, MDN with 2 mixtures, noted as MDN(2), MDN with 5 mixtures, noted as MDN(5), as comparisons.

Fig. 5.3 and Table 5.2 show the results obtained by different methods. From the results, we can see that the proposed models outperform the baseline model proposed in [Rojo et al., 2019]. Moreover, M1 and M2 can provide satisfying results even when the labeled data are scarce. The predicting accuracy is improved when the labeled data increases.

In contrast with other methods, the proposed models have better performance. Through the experiments, we also find that the proposed models, compared to other methods, besides the modeling accuracies, have the following advantages:

- Compared to the GP model, the proposed models are less computationally expensive;
- Compared to the MDN models, the proposed models are more computationally stable.

5.4/ CONCLUSION

In this chapter, we propose a VAE-based semi-supervised model for accurate indoor position recognition. In the unsupervised learning procedure, we use a Variational Autoencoder to learn a latent distribution with all the unlabeled data. For the supervised learning procedure, we design two predictors, one is deterministic and the other is stochastic. We utilize the latent distribution as the input to feed the predictor neural network so as to learn the final output. The advantage of doing so is that using the latent variables instead of using the original input can alleviate the overfitting problem.

For evaluating the proposed models, we choose a real-world dataset and conduct a series of the experiments with different amounts of labeled data to compare our model with other methods. The results show that the modeling accuracy is improved as the labeled portion increases. Meanwhile, the final results show that our method outperforms other existing methods as well.

6

Recognizing Indoor Location via End-to-End Learning

6.1/ Introduction

In the previous chapter, we propose to calculate the accurate user location by using the related WiFi fingerprints via the semi-supervised VAE model and the performances are significantly improved compared to the existing methods. However, as a semi-supervised learning model, the training process is not straightforward. Thus, in this chapter, we aim to develop a method which can be trained via end-to-end learning and achieves better performance.

To this end, we treat this problem as a supervised regression problem, whose input is WiFi RSSI values and whose output is the actual user location (latitudes and longitudes), as in the previous chapter. The difference is that, the in this chapter, we solve this problem directly via end-to-end learning instead of using semi-supervised learning. However, we have to deal with the same issues in Chapter 5, i.e., high dimensionality and noise.

For this reason, in contrast with the existing methods, based on the Information Bottleneck (IB) method [Tishby et al., 2000] and Variational Inference (VI), we propose a Variational Information Bottleneck (VIB)-based model [Alemi et al., 2017] in this chapter. This model consists of two sub-models, one is the encoder model used to compress the input, the other is the predictor model used to predict the target values. According to the Information Bottleneck theory, the encoder in our model is used to find a good latent representation of the input data for the related learning task so that the nuisance information in the original

input will be token out. Afterwards, the predictor utilizes the learnt latent representation as its input, instead of the original input, to predict the target values. Our model is an end-to-end deep learning model and scalable to large scale datasets, which makes it easy to train.

The main contributions in this chapter are summarized as follows:

- We devise a Variational Information Bottleneck model for computing accurate user location with WiFi fingerprint data;
- We vary the value of β of the proposed model to find the optimal value;
- In order to compare our method with other previous methods, we conduct a series of evaluation experiments.

6.2/ METHOD

Mathematically speaking, our goal is to map a very high dimensional source distribution, about 520, to a rather low dimensional target distribution, typically 2. However, the issue is that the dimension ”gap” between the two distributions is too large, which easily results in overfitting. Therefore, a better way to accomplish our task is to find a low dimension manifold to connect the input subspace and output space.

6.2.1/ MODEL SETUP

In our model, the input is the WiFi RSSI values x, the output is the coordinates of the user, y. To make the model more robust to noise, we use a set of probabilistic distributions such as $p(z|x)$ and $p(y|z)$ to describe the relationship between the variables instead of deterministic functions as opposed to conventional neural networks. Furthermore, in order to build the theoretical base for our model, we need to make the following assumptions first:

- **Assumption** 1: There exists a latent distribution of z governing both the input x and output y and consequently we have the information Markov chain: $x \rightarrow z \rightarrow y$.
- **Assumption** 2: The input x is solely sufficient enough to learn z, i.e., $p(z|x, y) = p(z|x)$.

- **Assumption** 3: The learnt latent representation z is solely sufficient enough to learn the output y, i.e., $p(y|x,z) = p(y|z)$.

We make the above assumptions based on the idea that the values of both the WiFi RSSIs and GPS coordinates are related to the real physical position of the users. Hence, either the WiFi RSSI values or the GPS coordinates has the sufficient information for calculating the real user physical position (which we denote it as the latent variable z). It means that we can use x to compute $p(z|x)$ (encoding step) and to compute $p(y|z)$ (predicting step). What's more, with the above assumptions, the derivations of our model will be easier.

Moreover, as it can be seen here, compared to the assumptions made in the VAE-based semi-supervised model in Chapter 5, we enhance the assumptions by adding Assumption 3, which allows us to develop a straightforward end-to-end model.

6.2.2/ MODEL

In a Maximum A Posteriori (MAP) modeling setting, the parameters of the model is related to not only the dataset but also the prior of the parameters:

$$p(\lambda|D) \propto p(D|\lambda)q(\lambda) \tag{6.1}$$

where D is the dataset, λ is the model parameters, $p(\lambda|D)$ is the posterior, $p(D|\lambda)$ is the likelihood and $q(\lambda)$ is the prior. Applying such a setting to our problem, then the prior of the latent representation z, $q(z)$ and the posterior $p(z|x)$ can both be represented by Gaussian distributions. In Variational Inference, $p(z|x)$ can be calculated via deep neural networks.

In Variational Autoencoders, one assumes that there is a latent distribution of z which can be used to reconstruct the original input x. Hence the information Markov chain for VAEs is $x \rightarrow z \rightarrow x'$, where x' is the reconstructed input. Accordingly, the loss function can be written as follow:

$$\mathcal{L}(D, w, \phi) = \mathbb{E}_{z\sim p_\phi(z|x)}[p_w(x|z)] - D_{KL}(p_\phi(z|x)||q(z)) \tag{6.2}$$

where ϕ is the parameters of the encoder network, w is the parameters of the decoder network, $q(z)$ is an uninformative prior of z, here we can use a standard Normal distribution

$\mathcal{N}(0, \mathbb{I})$.

Meanwhile, according to the Information Bottleneck principle [Tishby et al., 2000], [Tishby et al., 2015], let x be the input, y be the learning target and z be the representation, then we can have the following optimization objective:

$$\begin{aligned} &\max\ I(Z;Y) \\ &\text{s.t. } I(X;Z) \leq I_C \end{aligned} \tag{6.3}$$

where I denotes the mutual information, I_c is the information constraint.

In information theory, mutual information (MI) is used to measure the dependence between two random variables. The mutual information between two variables equals 0 if and only if the two variables are independent.

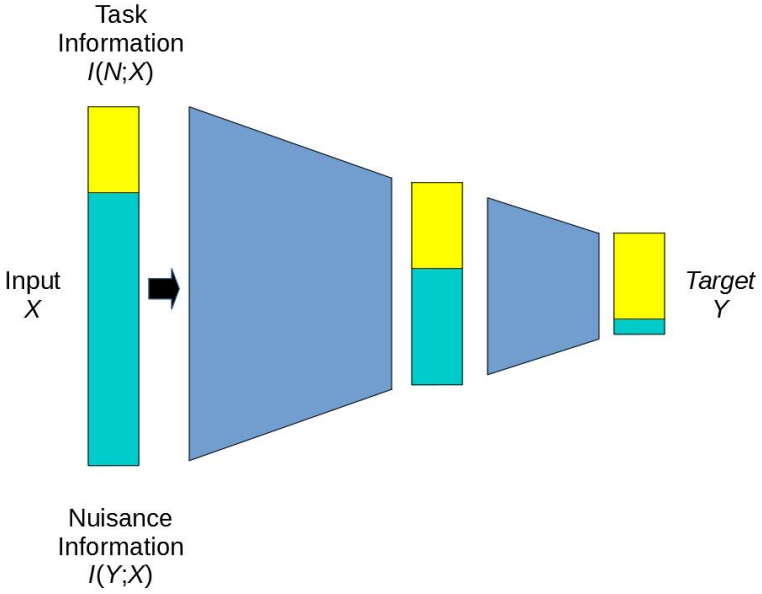


Figure 6.1. The information bottleneck.

Fig. 6.3 demonstrates the principle of information bottleneck. As it can be seen that for a multi-layer neural network, the nuisance information, which is not related to the learning task, is less when it is closer to the target.

Since it is tricky to solve Eq. (6.3) directly, we need to apply the Karush-Kuhn-Tucker

(KKT) conditions [Avriel, 2003] to Eq. (6.3), then we can cast the constrained optimization problem into an unconstrained optimization problem, as a result, we will have the following Lagrangian form of Eq. (6.3):

$$\mathcal{L}_{IB} = I(Z;Y) - \beta I(X;Z) \tag{6.4}$$

where $I(X;Z)$ is the upstream task used to compress the input, $I(Z;Y)$ is the downstream task used to predict the input, β is a Lagrangian multiplier controlling the trade off between the upstream task and downstream task.

6.2.2.1/ VARIATIONAL APPROXIMATION

However, Eq. (6.4) is still computationally prohibitive, thus we need to derive a variational approximation. To this end, first, we derive the variational lower bound for $I(Z;Y)$:

$$\begin{aligned}
I(Z;Y) &= \iint p(z,y) \log \frac{p(z,y)}{p(z)p(y)} dzdy \\
&= \mathbb{E}_{p(z,y)} \left[\log \left\{ \frac{p(z,y)}{p(z)p(y)} \right\} \right] \\
&= \mathbb{E}_{p(z,y)} \left[\log \left\{ \frac{p(z,y)}{p(z)} \right\} - \log \{p(y)\} \right] \\
&= \mathbb{E}_{p(z,y)} \left[\log \left\{ \frac{p(y|z)p(z)}{p(z)} \right\} - \log \{p(y)\} \right] \\
&\geq \mathbb{E}_{p(z,y)} [\, \log \{p(y|z)\} \,]
\end{aligned} \tag{6.5}$$

Now we derive the upper bound for $I(X;Z)$:

$$\begin{aligned}
I(Z;X) &= \iint p(z,x)\log\frac{p(z,x)}{p(z)p(x)}dzdx \\
&= \iint p(z,x)\log\left\{\frac{p(z,x)}{p(z)p(x)}\right\}dxdz \\
&= \iint p(z,x)\log\left\{\frac{p(z|x)q(z)}{p(z)q(z)}\right\}dxdz \\
&= \iint p(z,x)\left[\log\left\{\frac{p(z|x)}{q(z)}\right\} - \log\left\{\frac{p(z)}{q(z)}\right\}\right]dxdz \\
&= \iint p(z|x)p(x)\log\left\{\frac{p(z|x)}{q(z)}\right\} - \iint p(z|x)p(x)\log\left\{\frac{p(z)}{q(z)}\right\}dxdz \\
&= \mathbb{E}_x\left[D_{KL}(p(z|x)||q(z))\right] - D_{KL}\Big(p(z)||q(z)\Big) \\
&\leq \mathbb{E}_x\left[D_{KL}(p(z|x)||q(z))\right] \qquad (6.6)
\end{aligned}$$

Since our learning task is supervised, as opposed to VAEs and β-VAEs, we have this information Markov chain: $X \rightarrow Z \rightarrow Y$. As opposed to β-VAEs [Higgins et al., 2017], [Burgess et al., 2018], based on Eq. (6.3) and the assumptions we have made, we know that the latent variable z can be represented by x alone ($p(z|x,y) = p(z|x)$) and the output y can depend on y alone ($p(y|x,z) = p(y|z)$). For this reason, we can replace the term $p(x|z)$ in Eq. (6.2) with $p(y|z)$. As a result, now the original optimization objective Eq. (6.4) can be cast into a new optimization objective:

$$\begin{aligned}
&\underset{\theta,\,\phi}{\operatorname{argmax}}\ \mathbb{E}_D[\mathbb{E}_{p_\phi(z|x)}[\log p_w(y|z)]] \\
&\qquad s.t.\ \mathbb{E}_D\left[D_{KL}(p(z|x)||q(z))\right] \leq \epsilon \qquad (6.7)
\end{aligned}$$

where $D = \{x, y\}$ is the dataset, ϕ is the parameters of the encoder network, w is the parameters of the predictor network, ϵ is a positive constant with small value.

Based on Eq. (6.4) and Eq. (6.7), we can have the following lower bound:

$$I(Z;Y) - \beta I(Z;X) \geq \mathbb{E}_D\left[\mathbb{E}_{z\sim p(z|x)}\left[\log\{p(y|z)\}\right] - \beta D_{KL}(pp_\phi(z|x)||q(z))\right] \qquad (6.8)$$

Our purpose is to maximize Eq. (6.8), which is equivalent to minimizing the following loss

function:

$$\mathcal{L}(D,\theta,\phi) = \mathbb{E}_D\Big[\mathbb{E}_{z\sim p_\phi(z|x)}[-\log\{p_\theta(y|z)\}] + \beta D_{KL}(p_\phi(z|x)\|q(z))\Big] \tag{6.9}$$

Eq. (6.9) is the final loss function of our proposed model. In contrast with VAEs and β-VAEs, which are unsupervised learning models, whereas our model is an end-to-end supervised learning model. As shown in Fig. 6.2, $p_\phi(z|x)$ represents the encoder neural network and $p_w(y|z)$ represents the predictor neural network.

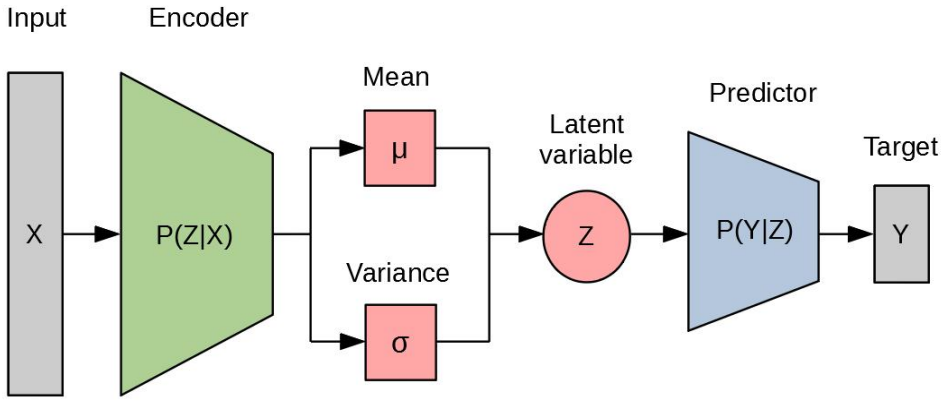


Figure 6.2. The structure of the VIB model.

6.2.2.2/ SOLVING MODEL

To solve Eq. (6.9), we need to adopt some special techniques. First, for computing the term $D_{KL}(p_\phi(z|x)\|q(z))$, we can use the reparameterization trick proposed in [Kingma et al., 2013] to make the parameters of the neural networks differentiable. In the reparameterization trick, the random distribution of z is decomposed as the combination of the standard deviation μ and the variance σ:

$$z = \mu_z + \sigma_z \odot \epsilon_z \tag{6.10}$$

where μ_z and σ_z can be calculated via the neural networks respectively, and the random noise ϵ_z can be sampled from a standard diagonal Normal distribution $\mathcal{N}(0,\mathbb{I})$.

Afterwards we need to calculate the term $\mathbb{E}_{z\sim p_\phi(z|x)}[p_w(y|z)]$. This term cannot be solved directly but we can use Monte Carlo method to compute it.

We adopt Monte Carlo sampling, and Eq. (6.9) becomes:

$$\mathcal{L}(D, w, \phi) = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\epsilon_z\sim p(\epsilon_z)}[p_w(y_n|f_\phi(x_n, \epsilon_z))] - \beta D_{KL}(p_\phi(z|x_n)||q(z)) \tag{6.11}$$

where N the total instance number, $f_\phi(x)$ is the same deterministic neural network used in the encoder to calculate the parameters of the distribution $p(z|x)$:

$$f_\phi(x) = \mu_z(x) + \sigma_z(x) \odot \epsilon_z \tag{6.12}$$

Note that β is a hyperparameter used to balance the encoding term and the predicting term so that it needs to be chosen very carefully.

6.2.2.3/ PREDICTING

In VAEs and β-VAEs, one can obtain new samples from an uninformative standard Gaussian first then use them as the input of the decoder. Whereas since our model is a supervised model, we use the samples from the conditional distribution, i.e., $p(z|x)$, to feed the predictor network, which is the same as the training procedure.

Algorithm 5 Algorithm

Input: X (input), Y (target)

Output: Y' (prediction)

while epoch $\leq$ max epoch **do**
  $\mu_z, \sigma_z \leftarrow E_\phi(X)$ ▷ $E_\phi(X)$: Encoder network
  $z \sim \mathcal{N}(\mu_z, \sigma_z)$ ▷ Sample latent codes
  $Y' \leftarrow F_y(z; w)$ ▷ F_y: Predictor network
  minimize loss function $\mathcal{L}(D, w, \phi)$ ▷ Eq. (6.9)
end while
return Y'

The overall algorithm is summarized in Algorithm 5.

6.3/ Experimental Results

6.3.1/ Dataset Description

For the validation, we use the UJIindoor dataset [Torres-Sospedra et al., 2014] whose input are 520 dimensional and each dimension represents a WAP. The RSSI values range from -110 dB to 0 dB when the WAPs are detected, otherwise the RSSI values are set to be 100. Also each RSSI vector corresponds to a pair of latitude and longitude as the geo-location label. In our experiments, we use scaled GPS coordinates values for computational convenience. The total instance number is about 20000. For Experiment 1 and Experiment 2, we use 80% of the dataset for training and the rest 20% as the test dataset. In Experiment 3, the training data number will vary.

6.3.2/ Model Implementation Details

Table 6.1 demonstrates the implementation details of our model. The encoder neural network includes of one hidden layer, and the dimension of the latent codes is set to be 5. In practice, we find that the latent dimension of 5 can be regarded as the Minimal Description Length [Hinton et al., 1994] for our task. The predictor is composed of three hidden layers. Each hidden layer has 512 units. Especially, in order to improve modeling generalization on test data, we can increase the model uncertainty. Hence we apply the Dropout technique [Dahl et al., 2013] to the hidden layers of the predictor. The optimizer for the model is Adam [Kingma et al., 2014a] and the learning rate is 10^{-3}.

6.3.3/ Experiment 1

In the loss function of the proposed model, the constant β is related to the constraint for the optimization, which is to balance the encoding error term $\mathbb{E}_{z\sim p_\phi(z|x)}[p_w(y|z)]$ and the prediction error term $D_{KL}(p_\phi(z|x_n)||q(z))$. A larger β value means the model tends to be more compressive for the input and less expressive for the output, and vice versa. Therefore, different β values can result in different modeling results.

To find the optimal β values, we will test different β values, ranging from 10^{-3} to 10^{-8}, for our model. From the results shown in Fig. 6.3, we can see that, when β is 10^{-6}, the

Table 6.1. Model Implementation Details

Sub-network	Layer	Parameter	Activation Function
Encoder	hidden layer	neuron number: 512; latent dimension: 5	ReLU
Predictor	hidden layer	neuron number: 512; dropout rate: 0.3	ReLU
Predictor	hidden layer	neuron number: 512; dropout rate: 0.3	ReLU
Predictor	hidden layer	neuron number: 512; dropout rate: 0.3	ReLU
Optimizer: Adam; learning rate: 10^{-3}.			

proposed model has the best performance. Thus, we will hereafter set β to 10^{-6} for the propose model in all following experiments.

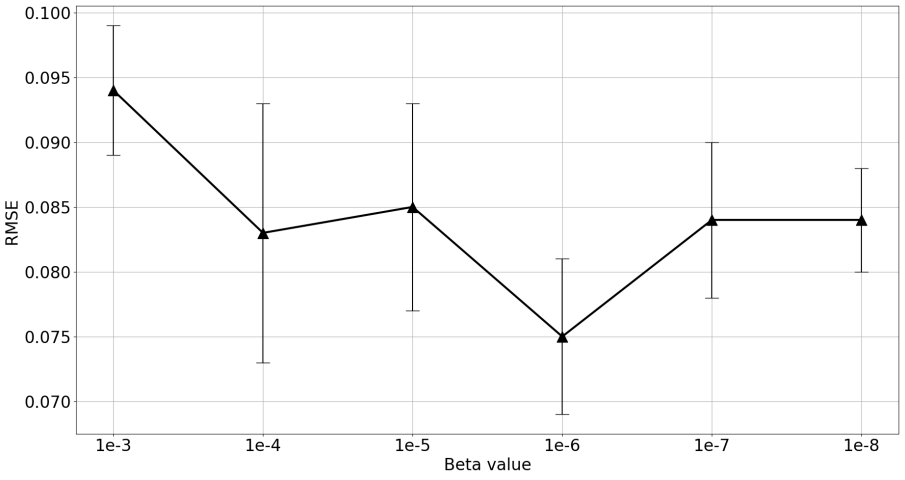
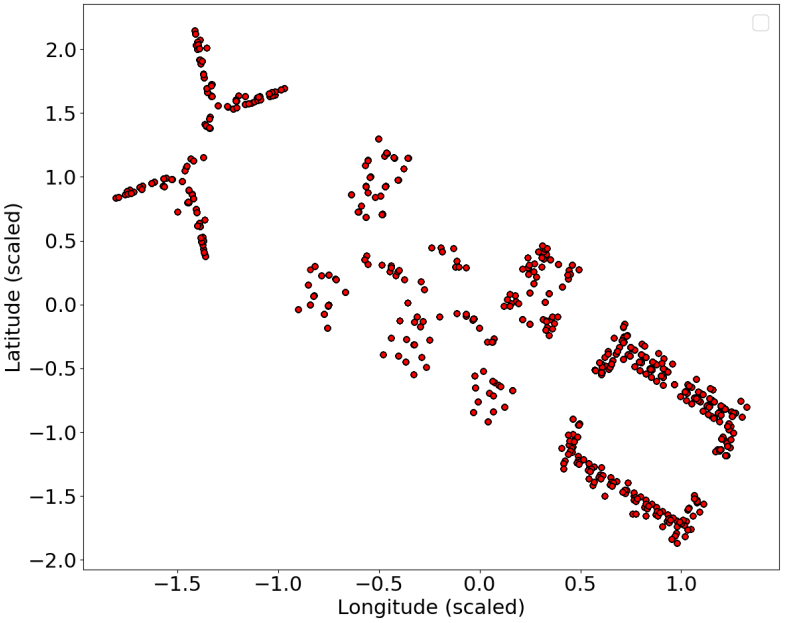
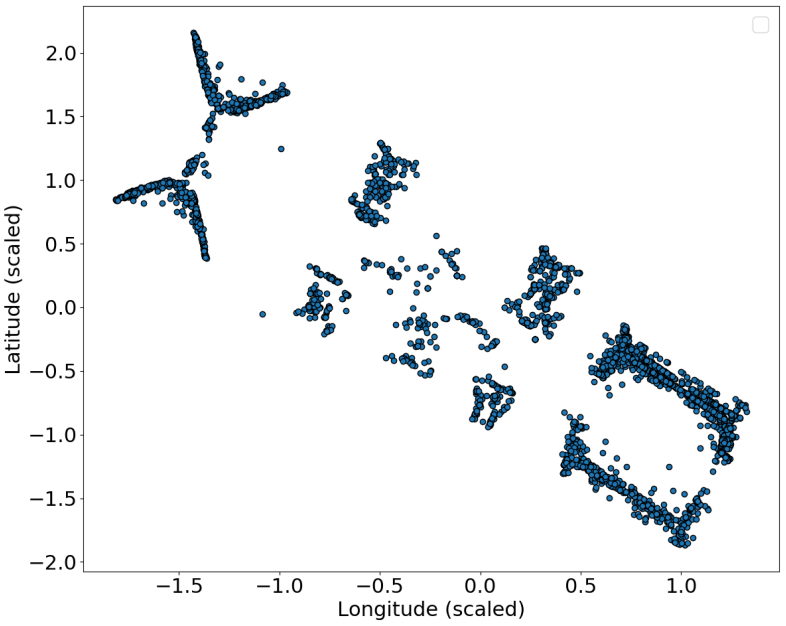


Figure 6.3. Results with respect to different β values.

Fig. 6.4 shows the ground truth and the testing result of our model. It can be seen the proposed model can calculate the location coordinates of the users accurately using the relevant WiFi fingerprints.

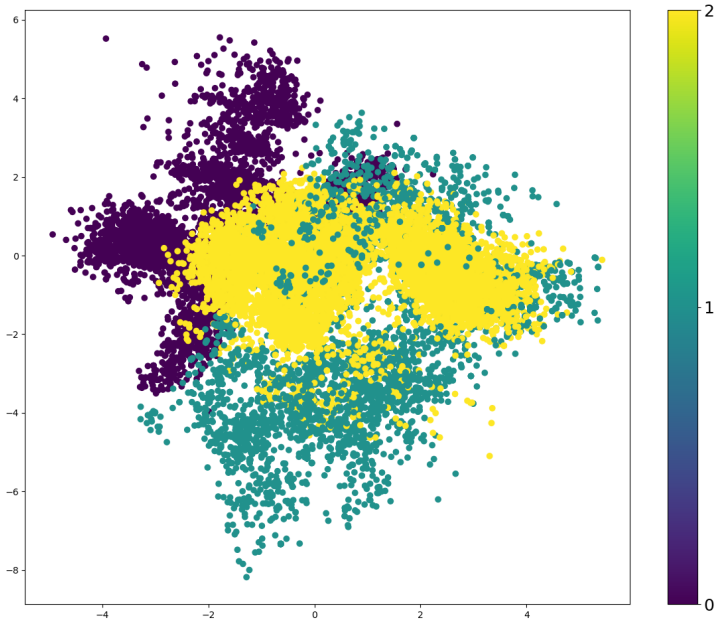


(a) Ground truth.

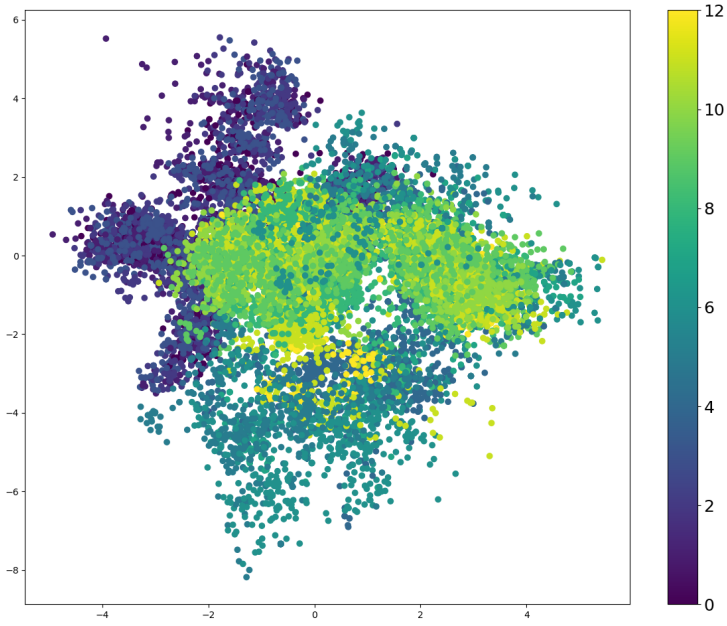


(b) Testing result.

Figure 6.4. Experimental result of the VIB-based model.



(a) Latent variables labeled with the building IDs, here shows the 2D projection.



(b) Latent variables labeled with the floor IDs, here shows the 2D projection.

Figure 6.5. Latent variables with dimension of 5, here shows the 2D projection.

In addition, Fig. 6.5 demonstrates how the latent distribution is related to the building IDs and floor IDs, respectively. And it verifies the assumptions we made before, i.e., the latent variable Z governs both the input X and the output Y.

Table 6.2. Comparison Results

(a) Results of the Baseline Models

Method	k-NN	GP	DT	RF
RMSE	0.092 ± 2e-3	0.252 ± 3e-3	0.112 ± 3e-3	0.087 ± 3e-3

(b) Results of the Advanced Models

Method	MDN-2	MDN-5	BNN	Semi-VAE	Proposed
RMSE	0.099 ± 3e-4	0.103 ± 3e-3	1.033 ± 4e-3	0.077 ± 4e-3	0.075 ± 6e-3

6.3.4/ EXPERIMENT 2

In order to show the advantages of our method, we run other methods proposed in the literature on the UJIindoor dataset. K-NN is used as the baseline model. The MDN-2 model refers to the Mixture Density Network (MDN) model with 2 mixed Gaussian distributions at the output layers. Similarly, the MDN-5 model is a MDN model with 5 mixed Gaussian distributions at the output layers. The Semi-VAE model is a semi-supervised Variational Autoencoder (VAE) model, which will be explained later. The overall results are shown in Table 6.2. We use the root mean squared errors (RMSE) as the evaluation metrics.

From the results, we can see that the proposed model has the best modeling performance. Also in practice we find that compared to our model, the Gaussian process model suffers from heavy computation load and the MDN models are unstable during the learning process.

6.3.5/ EXPERIMENT 3

According to our previous assumptions, as an alternative approach, we can also formulate a semi-supervised learning approach, the semi-supervised VAE model proposed in Chapter 5. To compare with the semi-supervised learning approach, we run our model and other baseline models on different portions of the labeled data. As the results shown in Fig. 6.6, we can see that once the labeled data used for the supervised learning procedure is more than 10% of the total training data, our method surprisingly has the best performance among all the methods.

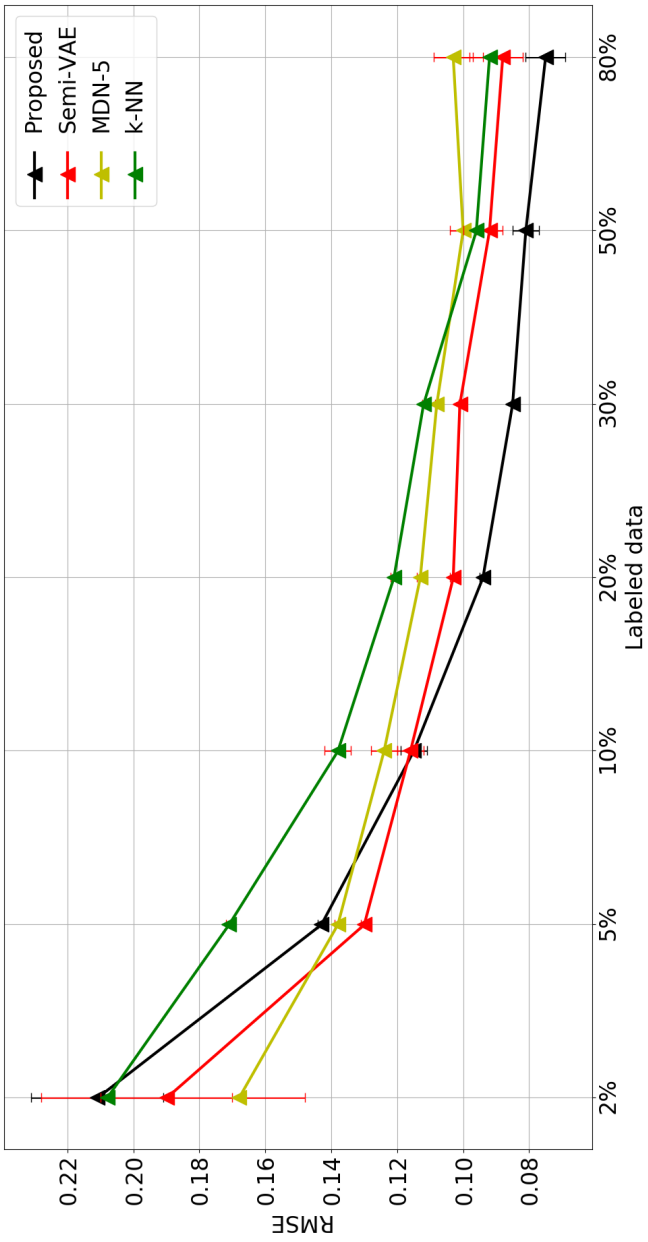


Figure 6.6. Results on different portions of the labeled data.

6.3.6/ DISCUSSION

Why the proposed method can outperform other deep learning methods? First, our problem can be regarded as a regression problem, and especially, the input (RSSI vectors) is relatively high dimensional and the target (GPS coordinates) is low dimensional. Thus, it causes the issue that the input has redundant information for the learning tasks. If we use a conventional neural network to solve this problem directly, the results will not be satisfying at all.

Mixture Density Networks (MDNs) and Bayesian Neural Networks (BNNs) handle this problem by inducing uncertainty into the models. The difference is that MDNs are MLE based method while BNNs are MAP based method. Surprisingly, BNNs have worse performance than MDNs on our tasks because the uncertainty of BNNs does not depend on the input data. Variational Autoencoders (VAEs) are originally designed as generative approaches to obtain new sample data. For out problem, we can use VAEs to learn the latent representation of the input data first. Then, this model can be trivially extended to be a semi-supervised model by using the pre-learned representation to obtain the final output.

However, in our study, we find that leveraging the Information Bottleneck method to this problem is a better option than the semi-VAE model. It is because that, with the Information Bottleneck method, we can view the original task as a constrained optimization problem. The optimization objective is the learning tasks and the constraint is the data representation. That's to say the Variational Information Bottleneck model is to directly find the optimal representation for the learning tasks, computing the output, whereas the semi-supervised VAE model is to find the representation to reconstruct the original input.

6.4/ CONCLUSION

Interpreting WiFi fingerprints into real user location via neural networks is a tricky problem. In this chapter, we combined the Information Bottleneck theory with Variational Inference to propose a novel deep learning model for WiFi fingerprint-based user location recognition. The proposed model consists of two neural networks, an encoder and a predictor. According to the Information Bottleneck theory, the encoder neural network

is to find an optimal representation of the data and mitigate the negative effect of the nuisance information for the learning tasks. The predictor neural network is to use the data representation to compute the final output. The main advantages of the proposed model is that it is scalable to large scale dataset, computationally stable and robust to noisy information. To evaluate our model, we run our model and other previous models on the real-world WiFi fingerprint dataset and the finally results verifies the effectiveness and show the advantages of our method compared to the existing approaches.

7

CONCLUSIONS AND PERSPECTIVES

7.1/ CONCLUSIONS

In this thesis, the research goal is studying human mobility through using the usage data collected from smartphone users. In order to have a comprehensive understanding, we have investigated user mobility from both outdoor and indoor aspects. Accordingly, we formulate the following tasks related to indoor and outdoor user mobility. Task 1 is discovering the daily mobility patterns of the users through using the collected GPS coordinate data; Task 2 is predicting the next time-point indoor user location with using the relevant WiFi fingerprint data; Task 3 is learning accurate indoor user location through using the relevant WiFi fingerprint data.

In order to accomplish the above tasks, we have investigated two types of approaches for learning user mobility, one type of approach is using GPS data and the other type is using WiFi fingerprint data. In particular, from a probabilistic perspective, we proposed the following solutions.

- For Task 1, we proposed a Dirichlet Process Gaussian Mixture Model (DPGMM)-based clustering algorithm to discover the daily user mobility patterns from the collected GPS coordinate data;
- For Task 2, we devise a hybrid sequential deep learning model, the Convolutional Mixture Density Recurrent Neural Network (CMDRNN), to predict the next time-point user location with the WiFi fingerprint data;
- For Task 3, we leveraged the idea of Variational Autoencoders (VAEs) to propose a VAE-based semi-supervised learning model for the indoor user location recognition

task. This model includes an encoder neural network, a decoder neural network and a predictor neural network;

- For Task 3, we further proposed an end-to-end deep learning model, the Variational Information Bottleneck (VIB) model for recognizing indoor user location.

The total contributions in this thesis are summarised as follows:

- Contribution 1: We first extract each daily trajectory from the whole user GPS dataset. Then, we use the Dirichlet Process Gaussian mixture model to estimate the probability density of each trajectory. Afterwards, we use the Kullback-Leibler divergence to measure the similarities between different trajectories. Finally, we use the computed similarities as the metrics to devise a automatic clustering algorithm to cluster the similar GPS trajectories into the same clusters.

- Contribution 2: In the CMDRNN model, we employ a 1D Convolutional Neural Network to detect the high dimensional input, a Recurrent Neural Network to represent the state transition in the time-series data, and a Mixture Density Network to sample the final output. With such design, our model can not only overcome the issue of high dimensionality but also the overfitting problem. For the validation, we conduct a series of experiments on a real-world dataset. In order to find the optimal hyper-parameters, we also compare different optimizers, different memory lengths and different mixture numbers.

- Contribution 3: In the VAE-based semi-supervised learning model, we use all the input data to learn a latent distribution in the unsupervised learning process. Then in the supervised process, we use the learnt latent distribution as the input of the new input for the predictor. Furthermore, we design two predictors, one predictor is a deterministic predictor and one predictor is a stochastic predictor.

- Contribution 4: We combines the Information Bottleneck method and Variational Inference to propose a Deep Variational Information Bottleneck model for user location recognition. This model consists of an encoder neural network and a decoder neural network. The advantage of the proposed model is that it is an end-to-end model, which makes it easier to train compared to the VAE-based model.

In order to validate the proposed methods, we conducted a series of experiments on several real-world datasets. The corresponding results show the effectiveness of our methods. We also compare the proposed methods with other existing methods, including conventional machine learning and deep learning methods. The final results suggest that our methods outperform other existing methods.

There are some remain works in this work. For instance, in terms of studying outdoor user mobility, we have devised a machine learning-based methods. But deep learning methods are known for being salable for large data scale, thus one can explore other advanced deep learning methods or combine probabilistic approaches with deep learning approaches to analyze GPS coordinate data.

7.2/ PERSPECTIVES

In this section, we will shed some lights on the possible future research. These research topics include, but are not limited to using other Usage data, improving sequential prediction, exploring other deep learning methods.

Using other Usage Data. In terms of applications, apart from GPS coordinate data and WiFi fingerprint data which we already used in our research, we can also take advantage of other kinds of smartphone usage data to study human behavior, such as application usage, cell Ids, call logs and battery usage, etc. These kinds of data enable researchers to investigate some other interesting research topics, such as smartphone application recommendation, travel destination recommendation and social relationship discovery. By doing so, we may be able to learn other types of user behavior so as to have holistic perspective of human behavior.

Improving Sequential Prediction. Our proposed methods have shown good performance in contrast with other existing methods but there are still room to improve them. For example, in terms of predicting indoor user location, one can leverage VAE or VIB models to develop latent recurrent deep learning models. n

Exploring other Deep Learning Methods. We can continue on focusing on improving probabilistic inference methods for deep learning models. There are some promising directions worthy digging into. For example, the performance of a Variational Autoencoder

can be enhanced by using the techniques like Normalising Flows and Autoregressive Flows to construct more complex posterior distributions. Instead of using mean-field assumption, one can use auxiliary variables to construct more complex posterior distributions [Maaløe et al., 2016].

Moreover, self-supervised representation learning has become an active research ares in recent years. Self-supervised learning is a kind of method whose loss function are supervised but it does not nedd labels. Based on mutual information estimation and maximization, some novel deep learning methods were proposed, for instance, Mutual Information Neural Estimation (MINE) [Belghazi et al., 2018], Contrastive Predictive Coding (CPC) [Oord et al., 2018] and Deep InfoMax (DIM) [Hjelm et al., 2019].

BIBLIOGRAPHY

[Abdi et al., 2010] Abdi, H., et Williams, L. J. (2010). **Principal component analysis**. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

[Agrawal et al., 1993] Agrawal, R., Faloutsos, C., et Swami, A. (1993). **Efficient similarity search in sequence databases**. In *International Conference on Foundations of Data Organization and Algorithms*, pages 69–84. Springer.

[Aldous, 1985] Aldous, D. J. (1985). **Exchangeability and related topics**. In *École d'Été de Probabilités de Saint-Flour XIII—1983*, pages 1–198. Springer.

[Alemi et al., 2017] Alemi, A. A., Fischer, I., Dillon, J. V., et Murphy, K. (2017). **Deep variational information bottleneck**. *5th International Conference on Learning Representations*.

[Ashbrook et al., 2003] Ashbrook, D., et Starner, T. (2003). **Using gps to learn significant locations and predict movement across multiple users**. *Personal and Ubiquitous computing*, 7(5):275–286.

[Avriel, 2003] Avriel, M. (2003). **Nonlinear programming: analysis and methods**. Courier Corporation.

[Baumann et al., 2018] Baumann, P., Koehler, C., Dey, A. K., et Santini, S. (2018). **Selecting individual and population models for predicting human mobility**. *IEEE Transactions on Mobile Computing*, 17(10):2408–2422.

[Bazzani et al., 2010] Bazzani, A., Giorgini, B., Rambaldi, S., Gallotti, R., et Giovannini, L. (2010). **Statistical laws in urban mobility from microscopic gps data in the area of florence**. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(05):P05001.

[Belghazi et al., 2018] Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., et Hjelm, D. (2018). **Mutual information neural estimation**. In *Proceedings of the 35th International Conference on Machine Learning*, pages 531–540.

[Bian et al., 2018] Bian, J., Tian, D., Tang, Y., et Tao, D. (2018). **A survey on trajectory clustering analysis**. *arXiv preprint arXiv:1802.06971*.

[Bishop, 2006] Bishop, C. M. (2006). **Pattern recognition and machine learning**. Springer Science+ Business Media.

[Bishop, Christopher M, 1994] Bishop, Christopher M (1994). **Mixture density networks**.

[Blei et al., 2006] Blei, D. M., Jordan, M. I., et others (2006). **Variational inference for dirichlet process mixtures**. *Bayesian analysis*, 1(1):121–143.

[Blei et al., 2017] Blei, D. M., Kucukelbir, A., et McAuliffe, J. D. (2017). **Variational inference: A review for statisticians**. *Journal of the American statistical Association*, 112(518):859–877.

[Bozkurt et al., 2015] Bozkurt, S., Elibol, G., Gunal, S., et Yayan, U. (2015). **A comparative study on machine learning algorithms for indoor positioning**. In *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, pages 1–8. IEEE.

[Brabham, 2013] Brabham, D. C. (2013). **Crowdsourcing**. Mit Press.

[Breiman, 1996] Breiman, L. (1996). **Bagging predictors**. *Machine learning*, 24(2):123–140.

[Burgess et al., 2018] Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., et Lerchner, A. (2018). **Understanding disentangling in β-vae**. *arXiv preprint arXiv:1804.03599*.

[Cao et al., 2007] Cao, H., Mamoulis, N., et Cheung, D. W. (2007). **Discovery of periodic patterns in spatiotemporal sequences**. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):453–467.

[Castro et al., 2013] Castro, P. S., Zhang, D., Chen, C., Li, S., et Pan, G. (2013). **From taxi gps traces to social and community dynamics: A survey**. *ACM Computing Surveys (CSUR)*, 46(2):1–34.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., et Bengio, Y. (2014). **Learning phrase representations using RNN**

encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

[Cho, 2016] Cho, S.-B. (2016). **Exploiting machine learning techniques for location recognition and prediction with smartphone logs**. *Neurocomputing*, 176:98–106.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., et Bengio, Y. (2014). **Empirical evaluation of gated recurrent neural networks on sequence modeling**. *arXiv preprint arXiv:1412.3555*.

[Cortes et al., 1995] Cortes, C., et Vapnik, V. (1995). **Support-vector networks**. *Machine learning*, 20(3):273–297.

[Cramariuc et al., 2016] Cramariuc, A., Huttunen, H., et Lohan, E. S. (2016). **Clustering benefits in mobile-centric wifi positioning in multi-floor buildings**. In *2016 International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–6. IEEE.

[Dahl et al., 2013] Dahl, G. E., Sainath, T. N., et Hinton, G. E. (2013). **Improving deep neural networks for lvcsr using rectified linear units and dropout**. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8609–8613. IEEE.

[Davis et al., 2011] Davis, R. A., Lii, K.-S., et Politis, D. N. (2011). **Remarks on some nonparametric estimates of a density function**. In *Selected Works of Murray Rosenblatt*, pages 95–100. Springer.

[Do et al., 2015] Do, T. M. T., Dousse, O., Miettinen, M., et Gatica-Perez, D. (2015). **A probabilistic kernel method for human mobility prediction with smartphones**. *Pervasive and Mobile Computing*, 20:13–28.

[Do et al., 2012] Do, T.-M.-T., et Gatica-Perez, D. (2012). **Contextual conditional models for smartphone-based human mobility prediction**. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, pages 163–172. ACM.

[Do et al., 2014] Do, T. M. T., et Gatica-Perez, D. (2014). **Where and what: Using smartphones to predict next locations and applications in daily life**. *Pervasive and Mobile Computing*, 12:79–91.

[Doersch, 2016] Doersch, C. (2016). **Tutorial on variational autoencoders**. *arXiv preprint arXiv:1606.05908*.

[Doucet et al., 2013] Doucet, A., De Freitas, N., Murphy, K., et Russell, S. (2013). **Rao-blackwellised particle filtering for dynamic bayesian networks**. *arXiv preprint arXiv:1301.3853*.

[Elman, 1990] Elman, J. L. (1990). **Finding structure in time**. *Cognitive science*, 14(2):179–211.

[Eren et al., 2019] Eren, L., Ince, T., et Kiranyaz, S. (2019). **A generic intelligent bearing fault diagnosis system using compact adaptive 1d cnn classifier**. *Journal of Signal Processing Systems*, 91(2):179–189.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et others (1996). **A density-based algorithm for discovering clusters in large spatial databases with noise.** In *Kdd*, volume 96, pages 226–231.

[Etter et al., 2013] Etter, V., Kafsi, M., Kazemi, E., Grossglauser, M., et Thiran, P. (2013). **Where to go from here? mobility prediction from instantaneous information**. *Pervasive and Mobile Computing*, 9(6):784–797.

[Ferris et al., 2007] Ferris, B., Fox, D., et Lawrence, N. D. (2007). **Wifi-slam using gaussian process latent variable models.** In *IJCAI*, volume 7, pages 2480–2485.

[Fuglede et al., 2004] Fuglede, B., et Topsoe, F. (2004). **Jensen-shannon divergence and hilbert space embedding**. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, page 31. IEEE.

[Gast, 2005] Gast, M. (2005). **802.11 wireless networks: the definitive guide**. " O'Reilly Media, Inc.".

[Gers et al., 1999] Gers, F. A., Schmidhuber, J., et Cummins, F. (1999). **Learning to forget: Continual prediction with lstm**.

[Gilks et al., 1995] Gilks, W. R., Richardson, S., et Spiegelhalter, D. (1995). **Markov chain Monte Carlo in practice**. Chapman and Hall/CRC.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., et Courville, A. (2016). **Deep learning**. MIT press.

[Graves et al., 2005] Graves, A., Fernández, S., et Schmidhuber, J. (2005). **Bidirectional lstm networks for improved phoneme classification and recognition**. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer.

[Greff et al., 2016] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., et Schmidhuber, J. (2016). **Lstm: A search space odyssey**. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

[Hähnel et al., 2006] Hähnel, B. F. D., et Fox, D. (2006). **Gaussian processes for signal strength-based location estimation**. In *Proceeding of robotics: science and systems*.

[Heckerman, 2008] Heckerman, D. (2008). **A tutorial on learning with bayesian networks**. In *Innovations in Bayesian networks*, pages 33–82. Springer.

[Hernández-Lobato et al., 2015] Hernández-Lobato, J. M., et Adams, R. (2015). **Probabilistic backpropagation for scalable learning of bayesian neural networks**. In *International Conference on Machine Learning*, pages 1861–1869.

[Hershey et al., 2007] Hershey, J. R., et Olsen, P. A. (2007). **Approximating the kullback leibler divergence between gaussian mixture models**. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–317. IEEE.

[Higgins et al., 2017] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., et Lerchner, A. (2017). **beta-vae: Learning basic visual concepts with a constrained variational framework.** *Iclr*, 2(5):6.

[Hinton et al., 2006] Hinton, G. E., et Salakhutdinov, R. R. (2006). **Reducing the dimensionality of data with neural networks**. *science*, 313(5786):504–507.

[Hinton et al., 1994] Hinton, G. E., et Zemel, R. S. (1994). **Autoencoders, minimum description length and helmholtz free energy**. In *Advances in neural information processing systems*, pages 3–10.

[Hjelm et al., 2019] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., et Bengio, Y. (2019). **Learning deep representations by mutual information estimation and maximization**. In *7th International Conference on Learning Representations*.

[Hoang et al., 2019] Hoang, M. T., Yuen, B., Dong, X., Lu, T., Westendorp, R., et Reddy, K. (2019). **Recurrent neural networks for accurate rssi indoor localization**. *arXiv preprint arXiv:1903.11703*.

[Hochreiter et al., 1997] Hochreiter, S., et Schmidhuber, J. (1997). **Long short-term memory**. *Neural computation*, 9(8):1735–1780.

[Hofmann-Wellenhof et al., 2012] Hofmann-Wellenhof, B., Lichtenegger, H., et Collins, J. (2012). **Global positioning system: theory and practice**. Springer Science & Business Media.

[Ibrahim et al., 2018] Ibrahim, M., Torki, M., et ElNainay, M. (2018). **Cnn based indoor localization using rss time-series**. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 01044–01049. IEEE.

[Jiang et al., 2012] Jiang, S., Ferreira, J., et González, M. C. (2012). **Clustering daily patterns of human activities in the city**. *Data Mining and Knowledge Discovery*, 25(3):478–510.

[Khetarpaul et al., 2011] Khetarpaul, S., Chauhan, R., Gupta, S., Subramaniam, L. V., et Nambiar, U. (2011). **Mining gps data to determine interesting locations**. In *Proceedings of the 8th International Workshop on Information Integration on the Web: in conjunction with WWW 2011*, pages 1–6.

[Kim et al., 2018] Kim, K. S., Lee, S., et Huang, K. (2018). **A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on wi-fi fingerprinting**. *Big Data Analytics*, 3(1):4.

[Kingma et al., 2014a] Kingma, D. P., et Ba, J. (2014a). **Adam: A method for stochastic optimization**. *arXiv preprint arXiv:1412.6980*.

[Kingma et al., 2015] Kingma, D. P., et Ba, J. (2015). **Adam: A method for stochastic optimization**. In Bengio, Y., et LeCun, Y., editors, *3rd International Conference on Learning Representations (ICLR)*.

[Kingma et al., 2016] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., et Welling, M. (2016). **Improved variational inference with inverse autoregressive flow**. In *Advances in neural information processing systems*, pages 4743–4751.

[Kingma et al., 2013] Kingma, D. P., et Welling, M. (2013). **Auto-encoding variational bayes**. *arXiv preprint arXiv:1312.6114*.

[Kingma et al., 2014b] Kingma, D. P., et Welling, M. (2014b). **Auto-encoding variational bayes**. In Bengio, Y., et LeCun, Y., editors, *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

[Kiukkonen et al., 2010] Kiukkonen, N., Blom, J., Dousse, O., Gatica-Perez, D., et Laurila, J. (2010). **Towards rich mobile phone datasets: Lausanne data collection campaign**. *Proc. ICPS, Berlin*, 68.

[Krogh et al., 2001] Krogh, A., Larsson, B., Von Heijne, G., et Sonnhammer, E. L. (2001). **Predicting transmembrane protein topology with a hidden markov model: application to complete genomes**. *Journal of molecular biology*, 305(3):567–580.

[Laurila et al., 2013] Laurila, J. K., Gatica-Perez, D., Aad, I., Blom, J., Bornet, O., Do, T. M. T., Dousse, O., Eberle, J., et Miettinen, M. (2013). **From big smartphone data to worldwide research: The mobile data challenge**. *Pervasive and Mobile Computing*, 9(6):752–771.

[Laurila et al., 2012] Laurila, J. K., Gatica-Perez, D., Aad, I., Bornet, O., Do, T.-M.-T., Dousse, O., Eberle, J., Miettinen, M., et others (2012). **The mobile data challenge: Big data for mobile computing research**. Technical Report.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., et Hinton, G. (2015). **Deep learning**. *nature*, 521(7553):436–444.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et others (1998). **Gradient-based learning applied to document recognition**. *Proceedings of the IEEE*, 86(11):2278–2324.

[Li et al., 2017] Li, D., Zhang, J., Zhang, Q., et Wei, X. (2017). **Classification of ecg signals based on 1d convolution neural network**. In *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6. IEEE.

[Liao et al., 2007] Liao, L., Patterson, D. J., Fox, D., et Kautz, H. (2007). **Learning and inferring transportation routines**. *Artificial Intelligence*, 171(5-6):311–331.

[Lin et al., 2005] Lin, D.-B., et Juang, R.-T. (2005). **Mobile location estimation based on differences of signal attenuations for gsm systems**. *IEEE transactions on vehicular technology*, 54(4):1447–1454.

[Lin et al., 2014] Lin, M., et Hsu, W.-J. (2014). **Mining gps data for mobility patterns: A survey**. *Pervasive and mobile computing*, 12:1–16.

[Loh, 2011] Loh, W.-Y. (2011). **Classification and regression trees**. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23.

[Lohan et al., 2017a] Lohan, E. S., Torres-Sospedra, J., Leppäkoski, H., Richter, P., Peng, Z., et Huerta, J. (2017a). **Wi-fi crowdsourced fingerprinting dataset for indoor positioning**. *Data*, 2(4):32.

[Lohan et al., 2017b] Lohan, E.-S., Torres-Sospedra, J., Richter, P., Leppkoski, H., Huerta, J., et Cramariuc, A. (2017b). **Crowdsourced wifi database and benchmark software for indoor positioning**. *Data set], Zenodo. doi*, 10.

[Lou et al., 2009] Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., et Huang, Y. (2009). **Map-matching for low-sampling-rate gps trajectories**. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 352–361.

[Lu et al., 2013] Lu, X., Wetter, E., Bharti, N., Tatem, A. J., et Bengtsson, L. (2013). **Approaching the limit of predictability in human mobility**. *Scientific reports*, 3:2923.

[Maaløe et al., 2016] Maaløe, L., Sønderby, C. K., Sønderby, S. K., et Winther, O. (2016). **Auxiliary deep generative models**. In *33rd International Conference on Machine Learning (ICML 2016)*.

[Martin Arjovsky et al., 2017] Martin Arjovsky, S., et Bottou, L. (2017). **Wasserstein generative adversarial networks**. In *Proceedings of the 34 th International Conference on Machine Learning, Sydney, Australia*.

[McInerney et al., 2013] McInerney, J., Zheng, J., Rogers, A., et Jennings, N. R. (2013). **Modelling heterogeneous location habits in human populations for location prediction under data sparsity**. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 469–478. ACM.

[Moon, 1996] Moon, T. K. (1996). **The expectation-maximization algorithm**. *IEEE Signal processing magazine*, 13(6):47–60.

[Muhlenbrock et al., 2004] Muhlenbrock, M., Brdiczka, O., Snowdon, D., et Meunier, J.-L. (2004). **Learning to detect user activity and availability from a variety of sensor data**. In *Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the*, pages 13–22. IEEE.

[Murphy, 2012] Murphy, K. P. (2012). **Machine learning: a probabilistic perspective**. MIT press.

[Neal, 2000] Neal, R. M. (2000). **Markov chain sampling methods for dirichlet process mixture models**. *Journal of computational and graphical statistics*, 9(2):249–265.

[Nowicki et al., 2017] Nowicki, M., et Wietrzykowski, J. (2017). **Low-effort place recognition with wifi fingerprints using deep learning**. In *International Conference Automation*, pages 575–584. Springer.

[Oord et al., 2018] Oord, A. v. d., Li, Y., et Vinyals, O. (2018). **Representation learning with contrastive predictive coding**. *arXiv preprint arXiv:1807.03748*.

[Papamakarios et al., 2017] Papamakarios, G., Pavlakou, T., et Murray, I. (2017). **Masked autoregressive flow for density estimation**. In *Advances in Neural Information Processing Systems*, pages 2338–2347.

[Patterson et al., 2003] Patterson, D. J., Liao, L., Fox, D., et Kautz, H. (2003). **Inferring high-level behavior from low-level sensors**. In *International Conference on Ubiquitous Computing*, pages 73–89. Springer.

[Peddemors et al., 2010] Peddemors, A., Eertink, H., et Niemegeers, I. (2010). **Predicting mobility events on personal devices**. *Pervasive and Mobile Computing*, 6(4):401–423.

[Pirozmand et al., 2014] Pirozmand, P., Wu, G., Jedari, B., et Xia, F. (2014). **Human mobility in opportunistic networks: Characteristics, models and prediction methods**. *Journal of Network and Computer Applications*, 42:45–58.

[Rasmussen, 2000] Rasmussen, C. E. (2000). **The infinite gaussian mixture model**. In *Advances in neural information processing systems*, pages 554–560.

[Reynolds, 2015] Reynolds, D. (2015). **Gaussian mixture models**. *Encyclopedia of biometrics*, pages 827–832.

[Rezende et al., 2015] Rezende, D., et Mohamed, S. (2015). **Variational inference with normalizing flows**. In *International Conference on Machine Learning*, pages 1530–1538.

[Rojo et al., 2019] Rojo, J., Mendoza-Silva, G. M., Cidral, G. R., Laiapea, J., Parrello, G., Simó, A., Stupin, L., Minican, D., Farrés, M., Corvalán, C., et others (2019). **Machine learning applied to wi-fi fingerprinting: The experiences of the ubiqum challenge**. In *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE.

[Ruder, 2016] Ruder, S. (2016). **An overview of gradient descent optimization algorithms**. *arXiv preprint arXiv:1609.04747*.

[Scellato et al., 2011] Scellato, S., Musolesi, M., Mascolo, C., Latora, V., et Campbell, A. T. (2011). **Nextplace: a spatio-temporal prediction framework for pervasive systems**. In *International Conference on Pervasive Computing*, pages 152–169. Springer.

[Schiller et al., 2004] Schiller, J., et Voisard, A. (2004). **Location-based services**. Elsevier.

[Sethuraman, 1994] Sethuraman, J. (1994). **A constructive definition of dirichlet priors**. *Statistica sinica*, pages 639–650.

[Song et al., 2019] Song, X., Fan, X., Xiang, C., Ye, Q., Liu, L., Wang, Z., He, X., Yang, N., et Fang, G. (2019). **A novel convolutional neural network based indoor localization framework with wifi fingerprinting**. *IEEE Access*, 7:110698–110709.

[Su et al., 2000] Su, W., Lee, S.-J., et Gerla, M. (2000). **Mobility prediction in wireless networks**. In *MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No. 00CH37155)*, volume 1, pages 491–495. IEEE.

[Tang et al., 2015] Tang, J., Liu, F., Wang, Y., et Wang, H. (2015). **Uncovering urban human mobility from large scale taxi gps data**. *Physica A: Statistical Mechanics and its Applications*, 438:140–153.

[Teh et al., 2005] Teh, Y. W., Jordan, M. I., Beal, M. J., et Blei, D. M. (2005). **Sharing clusters among related groups: Hierarchical dirichlet processes**. In *Advances in neural information processing systems*, pages 1385–1392.

[Tieleman et al., 2012] Tieleman, T., et Hinton, G. (2012). **Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude**. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[Tishby et al., 2000] Tishby, N., Pereira, F. C., et Bialek, W. (2000). **The information bottleneck method**. *arXiv preprint physics/0004057*.

[Tishby et al., 2015] Tishby, N., et Zaslavsky, N. (2015). **Deep learning and the information bottleneck principle**. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE.

[Torres-Sospedra et al., 2014] Torres-Sospedra, J., Montoliu, R., Martínez-Usó, A., Avariento, J. P., Arnau, T. J., Benedito-Bordonau, M., et Huerta, J. (2014). **Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems**. In *2014 international conference on indoor positioning and indoor navigation (IPIN)*, pages 261–270. IEEE.

[Torres-Sospedra et al., 2015] Torres-Sospedra, J., Montoliu, R., Trilles, S., Belmonte, Ó., et Huerta, J. (2015). **Comprehensive analysis of distance and similarity measures for wi-fi fingerprinting indoor positioning systems**. *Expert Systems with Applications*, 42(23):9263–9278.

[Trevisani et al., 2004] Trevisani, E., et Vitaletti, A. (2004). **Cell-id location technique, limits and benefits: an experimental study**. In *Sixth IEEE workshop on mobile computing systems and applications*, pages 51–60. IEEE.

[Wagner et al., 2013] Wagner, D. T., Rice, A., et Beresford, A. R. (2013). **Device analyzer: Understanding smartphone usage**. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 195–208. Springer.

[Wagner et al., 2014] Wagner, D. T., Rice, A., et Beresford, A. R. (2014). **Device analyzer: Large-scale mobile data collection**. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):53–56.

[Wu et al., 2008] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et others (2008). **Top 10 algorithms in data mining**. *Knowledge and information systems*, 14(1):1–37.

[Xiao et al., 2010] Xiao, X., Zheng, Y., Luo, Q., et Xie, X. (2010). **Finding similar users using category-based location history**. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 442–445.

[Yang et al., 2019] Yang, T., Cappelle, C., Ruichek, Y., et El Bagdouri, M. (2019). **Online multi-object tracking combining optical flow and compressive tracking in markov decision process**. *Journal of Visual Communication and Image Representation*, 58:178–186.

[Yavaş et al., 2005] Yavaş, G., Katsaros, D., Ulusoy, Ö., et Manolopoulos, Y. (2005). **A data mining approach for location prediction in mobile environments**. *Data & Knowledge Engineering*, 54(2):121–146.

[Ye et al., 2012] Ye, J., Dobson, S., et McKeever, S. (2012). **Situation identification techniques in pervasive computing: A review**. *Pervasive and mobile computing*, 8(1):36–66.

[Yiu et al., 2015] Yiu, S., et Yang, K. (2015). **Gaussian process assisted fingerprinting localization**. *IEEE Internet of Things Journal*, 3(5):683–690.

[Yu et al., 2017] Yu, C., Liu, Y., Yao, D., Yang, L. T., Jin, H., Chen, H., et Ding, Q. (2017). **Modeling user activity patterns for next-place prediction**. *IEEE Systems Journal*, 11(2):1060–1071.

[Zhang et al., 2018] Zhang, C., Bütepage, J., Kjellström, H., et Mandt, S. (2018). **Advances in variational inference**. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026.

[Zhao et al., 2019] Zhao, J., Mao, X., et Chen, L. (2019). **Speech emotion recognition using deep 1d & 2d cnn lstm networks**. *Biomedical Signal Processing and Control*, 47:312–323.

[Zheng et al., 2010a] Zheng, V. W., Zheng, Y., Xie, X., et Yang, Q. (2010a). **Collaborative location and activity recommendations with gps history data**. In *Proceedings of the 19th international conference on World wide web*, pages 1029–1038.

[Zheng, 2015] Zheng, Y. (2015). **Trajectory data mining: an overview**. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29.

[Zheng et al., 2008] Zheng, Y., Li, Q., Chen, Y., Xie, X., et Ma, W.-Y. (2008). **Understanding mobility based on gps data**. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 312–321.

[Zheng et al., 2010b] Zheng, Y., et Xie, X. (2010b). **Learning location correlation from gps trajectories**. In *2010 Eleventh International Conference on Mobile Data Management*, pages 27–32. IEEE.

[Zheng et al., 2011] Zheng, Y., et Xie, X. (2011). **Learning travel recommendations from user-generated gps traces**. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(1):1–29.

[Zheng et al., 2009] Zheng, Y., Zhang, L., Xie, X., et Ma, W.-Y. (2009). **Mining interesting locations and travel sequences from gps trajectories**. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM.

[Zhou, 2012] Zhou, Z.-H. (2012). **Ensemble methods: foundations and algorithms**. CRC press.

List of Figures

List of Tables

A

Related Publications

Journal:

- **Weizhu QIAN**, Bowei Chen,Yichao Zhang, Guanghui Wen, and Franck Gechter, *Multi-Task Variational Information Bottleneck.* (under review), IEEE Transactions on Cybernetics, IEEE.
- **Weizhu QIAN**, Fabrice Lauri and Franck Gechter, *Supervised and Semi-supervised Deep Probabilistic Models for Indoor Positioning Problems* (accepted), Neurocomputing, Elsevier.

Conferences:

- **Weizhu QIAN** and Franck Gechter, *Variational Information Bottleneck Model for Accurate Indoor Position Recognition* (accepted), In the proceeding of the 25th International Conference on Pattern Recognition (ICPR 2020), 2020, Milan, IEEE.
- **Weizhu QIAN**, Fabrice Lauri and Franck Gechter, *A Probabilistic Approach for Discovering Daily Human Mobility Patterns with Mobile Data*, In the proceeding of the 18th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2020), p457–p470, 2020, Lisboa, Springer.
- **Weizhu QIAN**, Franck Gechter and Fabrice Lauri, *Comparisons of Machine Learning Algorithms on Smartphone Energy Consumption Modeling Issue Based on Real User Context Data*, (In the proceeding of the 11th International Conference on Information, Process,and Knowledge Management eKNOW 2019).

- **Weizhu QIAN** and Franck Gechter, *Modeling Smartphone Energy Consumption Based on User Behavior Data.* In the proceeding of the 5th International Conference on Computational Science and Computational Intelligence (CSCI'18), Las Vegas, 2018.

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/