東京大学大学院新領域創成科学研究科
社会文化環境学専攻

2021 年度
修　士　論　文

Location prediction of sparse mobility data based on federated transfer learning
連合転移学習に基づくスパースモビリティデータの位置予測

2021 年 7 月 9 日提出
指導教員　柴崎　亮介　教授
副指導教員　瀬崎　薫　教授

徐　　立強
Xu, LiQiang

Abstract

Human mobility prediction is a key problem in urban computing, which could be beneficial to developing smart city applications. Short-term human mobility prediction could be utilized to provide advertisement and above it, long-term human mobility prediction will help location-based AI applications in smart city in the future. Nowadays we could have obtained a relatively high prediction accuracy of human mobility prediction when we have huge users and high time resolution. Unfortunately, there are still some issues reserved when we predict sparse human mobility data or use privacy GPS data. As a sequential problem, human mobility prediction has been studied by Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM). Sparse data means it lack of information, so it is difficult to generate an effective model. Some studies attempt to incorporate spatiotemporal factors into the neural network or use attention mechanism to solve the problem caused by the sparsity of human mobility data. To some extent, these studies improve the performance of prediction of sparse human mobility data. However, few studies devote to improving accuracy of sparse human mobility data by help of dense human mobility data. Another problem is privacy issue. Although more and more human mobility data is collected in smart phones, use of these data is limited and companies couldn’t use the data collaboratively with others due to privacy problem, which results in "isolated islands" issue. Federated learning is privacy-preserving model training in heterogeneous, distributed networks, which could help companies collaborate to train a collective model that they can store their raw data locally and needn’t to exchanged data with each other. In order to solve these problems, this study utilizes transferring learning to adapt the domain from dense data to sparse data to help improve performance of prediction of sparse human mobility data and uses federated learning to train a collective model among companies without exchange of data. Combined with the two technologies, this study proposes a federated transfer learning model for modeling sparse human mobility data without exchange of raw data. This study uses two real-world GPS datasets that one is sparse dataset and another is dense dataset to model human mobility data. Result shows this model has great performance on prediction of sparse human mobility data by the help of dense data without exchange privacy.

Key words: Long-term human mobility prediction, Sparse human mobility data, Federated learning, Transfer Learning, Domain Adaptation, GPS

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Ryousuke Shibasaki, who supports us to conduct studies by our interests and freedom. In each group meeting, he listened carefully to our report and he gave me so many advices and guidance every time, which makes me realize my deficiency and problems so that I could keep improving myself in researches and become more confident. I am also grateful to my co-advisor Prof. Kaoru Sezaki, who also gave me practical direction on the research and thesis and it help me solve many research problems.

Then, thanks for my associate Professor Xuan Song, who shared a lot of research directions for us and kindly talk with me about my research and daily life. Of course, he also gave me many suggestions about my studies. Thanks to Researcher Zipei Fan, who has been my tutor for two years and he taught me so much from research direction to programing skills. I have at least one meeting with he every week, he taught me lots of methodologies and programing skills and gave me lots of practical advice. He also helps me in daily life and let me know more about Japan. I also thanks to research Renhe Jiang, who also gave me lots of useful proposals and research directions.

In addition, I thank all my friends in Shibasaki Lab: Thanks to Zhiwen Zhang, Yinhao Liu, HaoyuWang, Hang Yin, Lifeng Lin, Chuang Yang, Zekun Cai, Wenjin Li, Yanxiu Jin, whom make me enjoy life and study in the University of Tokyo. Thanks to senior student Jinyu Chen, Yuhao Yao, Zhaonan Wang, Dou Huang, Tianqi Xia who also help me in daily life and studies.

Thanks to my parents abroad who gave birth to me and always care about me even I'm abroad. Thanks for my father talking with me about my and future and always supports my decision. Thanks for my mother that always talks with me about my daily life in Japan. I am grateful to all my other relatives for their concern.

Thanks for my friends in undergraduate course, who care me and talk me about many researches contents and gave me many advice even they are far away from me now.

The coronavirus started from 2019 and still continue in 2021 now, which changed too much about my life. Since I come back Japan from spring of 2020, I have stay in Japan more than one and half years without go home, China. Under such circumstances, I took my course online and conducted my researches mainly at home. In such a circumstance almost alone in most time, I thank all people who care me and talk with me, which helps me conquer all obstacles to finish this thesis. I sincerely hope the epidemic could end quickly and the future will be more brilliant.

Contents

Abstract i
Acknowledgements iii
Chapter 1. Introduction 1
1.1 Background 1
1.2 Related works 3
1.3 Outline 5
Chapter 2. Methodology 7
2.1 Deep Neural Network 7
2.1.1 RNN 7
2.1.2 LSTM 8
2.1.3 GAN 9
2.1.4 Attention mechanism 10
2.2 Transfer learning 11
2.2.1 Measure criterion 11
2.2.2 Model pre-training transfer method 13
2.2.3 Domain Adaptation 13
2.3 Federated learning 17
2.3.1 Categories 17
2.3.2 FedAvg 18
Chapter 3. Problem statement 19
Chapter 4. Data pre-processing 22
4.1 Dataset 22
4.2 Data pretreatment 23
4.3 trajectory cluster 24
Chapter 5. Model 26
5.1 Activation function 26
5.2 Baseline 28
5.3 Federated learning model 30
5.4 Transfer learning model 31
5.5 Federated transfer learning model 32
Chapter 6. Experiment and analysis 35

6.1 Measure criterion 35
6.2 Experiment setups 36
6.3 Result of location prediction 37
6.4 Analysis 39
Chapter 7. Conclusion and future work 41
Reference 43

List of Figures

Figure 1: Architecture of this research **2**

Figure 2: Vanilla RNN **7**

Figure 3: Internal structure of LSTM **9**

Figure 4: Structure of GAN **10**

Figure 5: Attention sequence to sequence **11**

Figure 6: Network of DANN **16**

Figure 7: Categorization of federated learning **17**

Figure 8: 1600 common anchors in Tokyo **25**

Figure 9: Sigmoid function **26**

Figure 10: Tanh function **27**

Figure 11: Relu function **27**

Figure 12: Architecture of Flashback **30**

Figure 13: Federated learning process **31**

Figure 14: Transfer learning model **32**

Figure 15: Federated transfer learning model **34**

Figure 16: Loss comparison on basic model **37**

Figure 17: loss comparison on federated transfer learning model **38**

List of Table

Table 1: Statistics of the datasets **23**

Table 2: Adjacent Cluster Center Distance Statistics **24**

Table 3: Location prediction performance on different datasets with different models **38**

Chapter 1. Introduction

1.1 Background

Location prediction is a key problem in human mobility modeling and it is related to the design of location-based services, which could be beneficial to developing smart city applications. Virtually, many applications have utilized users' current location to provide service to users, such as real-time advertising push. However, due to instability and inaccuracy of user's future location and risk of leveraging user's privacy, although it is more important and attractive, it's difficult to utilize user's future location to provide service or product.

Location prediction could be divided into large-scale human cluster prediction and personal prediction. The former could be used to plan large-scale events, traffic and notify users. Individual predication will boost mobile advertising and help location-based AI applications in smart city, such as smart assistants and smart homes.

Nowadays we could have obtained a relatively high prediction accuracy when we have huge users and high time resolution. However, there are some problems when we predict human mobility. Firstly, the shortage of data. Virtually, many companies don't have so much users. Even though some companies have huge users, they still couldn't collect all users' data. Secondly, low quality of data and low-frequency sampling. The data collected may include some inaccurate data, these data could influence our model. Another problem is low-frequency sampling. Sampling interval could be too long, then the data will be too sparse. Sparse data means it lack of information, so it is difficult to generate an effective model.

With the rapid development of Internet of Things (IoT) technology, many organizations could have collected huge number of trajectories data, but to protect users of privacy, they will be unwillingness to share their data. Last problem is that different users have different

sizes of data and their data features are greatly different. Personalization should be considered when designing the network.

In many Locations Based Social Networks (LBSNs), a user's mobility trace is stored as a sequence of check-ins, where each check-in represents the user's presence at a specific Point of Interests (POIs) such as a restaurant or a gym, at a specific time. However, these check-in data is so sparse that it's hard to generate a model just by it. On another side, some companies, such as docomo in Japan, have huge dense data. If we could utilize these dense data to help this kind of sparse social-network data to predict, it will have a better result. However, the two parts may not want to share their data. Fortunately, some recent privacy-preserving technologies, such as federated learning [1] could help us solve this problem, it needn't us exchange two sides' data, just gradient when training. Then we can generate a predict model.

Although some technologies could help two companies collaboratively train models without sharing data. Another problem is that two sides' data could be greatly different. In machine learning field, we can think the two sides' data are in two domains [2]. If we merely combine them without other measures, it's still hard to have a great result. To solve this problem, we could use domain adaptation to align data when training the model. Domain adaptation is a field that aims at mapping the data distributed in different source and target domains to a feature space to make it as close as possible in that space [3].

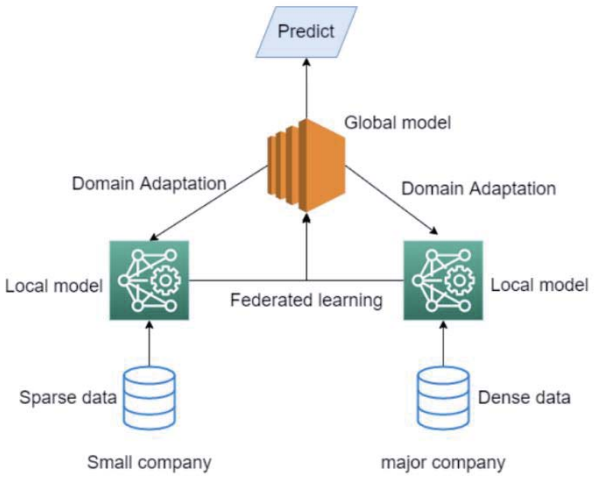


Figure 1: Architecture of this research

In summary, in this study, I make the following key contributions:

- We apply federated learning method based on FedAvg [4] on human mobility prediction between different datasets without sharing their data so that it can protect two parts' privacy when model training.

- We utilize two transfer learning methods that use dense data to help sparse data to predict users' next location and the two methods both have a relatively great performance

- We design a federated transfer learning model for modeling sparse human mobility data, which compared transfer learning and federated learning method. This model improves accuracy of prediction of sparse human mobility data and doesn't reveal privacy.

1.2 Related works

Location prediction is a key problem in human mobility modeling and it is a popular research issue in many fields, including traffic engineering [5,6], computer vision [7,8], ride-sharing [9] and emergency management [10]. Traditional methods of human mobility data prediction need various mobility features like hand-craft features, including historical visit counts [11,12] or uses graph embedding techniques to obtain automatically-learnt features [13]. Some methods attempt to predict future movements patterns by searching for similar trajectories in the past trajectories [14], integrating heterogeneous data sources [15] and modeling periodical mobility patterns [16]. Besides, factorization models are also used to solve location prediction/recommendation problems [17]. In addition, Markov Chains are widely used for sequential prediction [18]. FPMC is extended to the location prediction problem by considering spatial constraints [19] in building the transition matrices

Compared with traditional method, recurrent neural network (RNNs) has shown that it is a successful tool to model sequential data [20]. In order to handle sparse and incomplete

sequences, some studies consider temporal factors as additional inputs to the RNN units [21]. The most popular method to incorporate spatiotemporal factors into RNNs is that adds the spatiotemporal distances between check-ins [22,23]. Although these methods attempt to model sparse human mobility data, they couldn't fully utilize the rich historical spatiotemporal contexts. In order to make full use of spatiotemporal contexts, some studies add attention mechanism into RNNs [24] to consider spatial-temporal factors in human mobility data.

Transfer learning is aimed at transferring knowledge from existing domains to a new domain [25]. In transfer learning problems, the domains are greatly different but relevant, making transfer knowledge become possible. The key idea of transfer learning is reducing the distribution divergence between different domains. There are two mainly methods in transfer learning, instance reweighting and feature matching. Some studies reuse samples from the source domain according to some weighting technique [26,27]. Then some feature matching method utilizes learn subspace by exploiting the subspace geometrical structure [28] or align distribution to reduce marginal or conditional divergence between different domains [29]. Compared with these traditional methods, recently deep transfer learning is widely used and proves efficiency in many fields [30,31].

With rapid development of smart devices, more and more data are collected and privacy issue becomes an important problem. In 2016, the European Union proposed a General Data Protection Regulation [33] to protect privacy in order to grant the users' right, which asks companies to clarify where and what data they have collected and used that could avoid arbitrary use of data.

However, individual data could be used to help development of society and improve service. Therefore, in order to make full use of personal data, some studies are centralized on privacy preserving methods. Differential privacy [34], which provides a mathematical definition of privacy, has been deployed by Apple to protect the privacy of iOS users. Secure multi-party computation (MPC) is a subfield of cryptography aims at creating methods for parties to jointly compute a function over their inputs while keeping those inputs private. However, most of these methods sacrifice too much of the utility of the

data in order to better protect privacy. Therefore, they will greatly reduce prediction accuracy.

Federated learning is privacy-preserving model training in heterogeneous, distributed networks and it is firstly proposed by Google [32]. The motivation of it is solving problem data isolation and protect data privacy and security. In practical application, it will be used in smart phones of every user or across different organizations. Some studies focus on federated multi-task [35] and personalized federated learning [36]. Meta learning is also combined with federated learning to help federated learning initialization [37]. One problem in federated learning is that data could be non-iid, some studies attempt to solve this problem [38]. Communication and cost couldn't be ignored in updates in federated learning, some studies optimize process of federated learning method [39].

It's difficult to combine federated learning and transfer learning because in most transfer learning method, domain alignment occurs in every iteration, which means that we must mix different domain's data and it is conflict with federated learning. Fortunately, just recently, some studies find some method that could combine the two technologies [40].

1.3 Outline

The paper is structured as follows:

- Chapter 1 introduces background and related work about this study
- Chapter 2, introduces some deep neural network, basic knowledge and common algorithms of transfer learning and federated learning.
-
- Chapter 3 describes problem of prediction of human mobility data and defines some mathematical expressions
- Chapter 4 introduces two datasets, some pretreatment on two datasets and trajectory cluster method. This chapter is prepared for experiment in later chapters.

- Chapter 5 describes some activation function that is used in model. This chapter introduces baseline model and proposes federated learning and transfer learning model based on baseline model. On top of it, I propose a federated transfer learning model that could have a great performance on sparse human mobility data without privacy disclosure.

- Chapter 6 introduces measure criterion in model evaluation and experiment by federated learning method, transfer learning method, final mix method. Based on result of experiment, this chapter makes some analysis.

- Chapter 7 concludes this thesis by emphasizing the contributions and indicates the current deficiency of this study. At last, this chapter introduces some future work that could improve experiment results

Chapter 2. Methodology

2.1 Deep Neural Network

Nowadays, researches most use deep neural network (DNN) to complete prediction task in many fields, including computer vision, Natural language Processing and human mobility data. Most common sequence prediction network is recurrent neural network (RNN) and Long Short-Term Memory (LSTM). Attention mechanism is used to improve performance of network.

2.1.1 RNN

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.

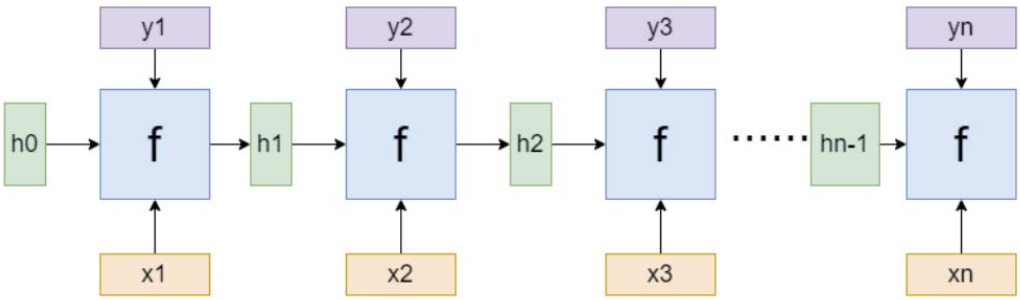


Figure 2: Vanilla RNN

$$h_t = f(U_{x_t} + W_{h_{t-1}+b}) \quad (1)$$

$$y_t = g(V_{s_t} + c) \quad (2)$$

The input is sequence x1, x2, x3… and initialization value h0. In this thesis, the input sequence is Past/most recent trajectory location and output is next trajectory location. Formula (5) is the calculation formula of the hidden layer, f is the activation function

generally using tanh function, U is the weight matrix from the input layer x_{t-1} to the hidden layer h_t, W is the weight matrix from the hidden layer h_{t-1} at the last moment to the hidden layer h_t at the current moment, b is the deviation vector.

Formula (6) is the calculation formula of the output layer, g is the activation function generally using SoftMax function, V is the weight matrix between the hidden layer h_t, and the output layer y_t, c is the deviation vector, and this layer is a fully connected layer. The hidden-layer node h_t could be considered as a unit of memory that captures information from all previous moments.

Sequential data is difficult to process with a basic neural network and RNN solves this problem because the output of the neuron can be transmitted to itself at the next moment. RNN can capture long-term dependencies of sequential information. Unfortunately, when the sequence is too long, especially when the sequence is over 20 words, its performance will degrade rapidly. In many previous studies, RNN has a great performance in short-term prediction, such as prediction in one day. However, when we predict long-term human mobility pattern, it couldn't have a great result.

2.1.2 LSTM

The parameters of the time dimension are shared, so when the sentence is long, the gradient will disappear in the process of back propagation. LSTM improves this problem to some extent.

In LSTM, there are three gates to control model, input gate, forget gate and output gate. The three gates' activation function is sigmoid function. Gate closes when the input is less than zero.

The formula of gates is

$$G = \sigma(W(x_t; h_{t-1}) + b) \quad (3)$$

W is parameter matrix; b is bias matrix and σ is activation function.

The formula calculating memory cell is:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot tanh(W_c(x_t; h_{t-1}) + b_c) \quad (4)$$

The output of hidden state is:

$$h_t = o_t \cdot tanh(C_t) \quad (5)$$

The memory cells of LSTM can store previous information, and the memory gate controls the preservation of past information, because LSTM can retain memories for longer than RNN.

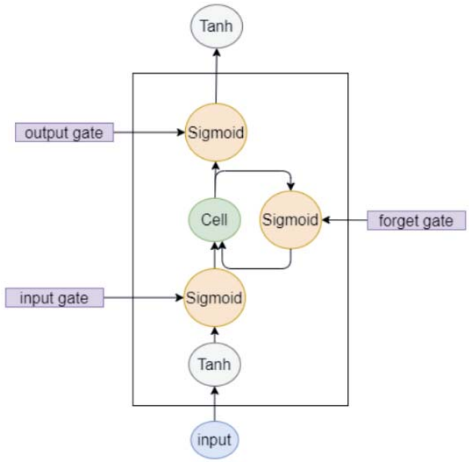


Figure 3: Internal structure of LSTM

2.1.3 GAN

Generative Adversarial Network (GAN) is one kind of generative model. It usually has a generator and discriminator.

The generator is used to sample random noise as an input from any given distribution, and the goal of the generator is to fool the discriminator by making the generated input as close as possible to the distribution of the real data. The discriminant determines which distribution the real data and the sample generated by the generator come from. Generators and discriminators update iteratively by continually adversarial learning. Finally, the generator can produce data with a distribution similar to the real data, while the discriminators cannot determine whether the input sample is real or false, thus achieving a Nash equilibrium.

P represents the distribution of real data, and P represents the distribution of data

generated by the generator. The optimization formula of the model can be expressed as:

$$min_G max_D E_{x\sim P_{data}} log[D(x)] + E_{z\sim p_z} log[1 - D(G(z))] \quad (6)$$

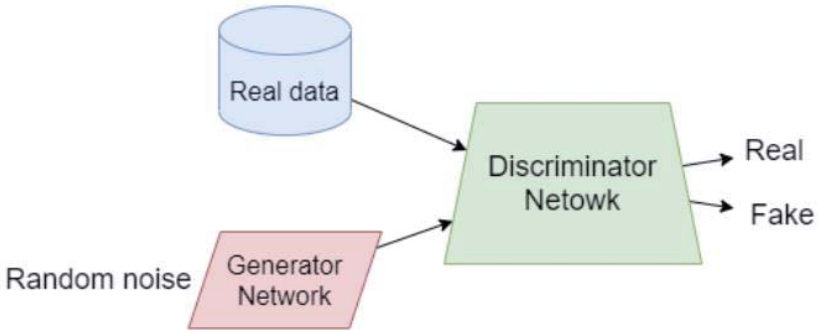


Figure 4: structure of GAN

2.1.4 Attention mechanism

The mechanism of Attention, at a superficial level, fits its name perfectly. Its core logic is focusing on the key rather than focusing on the whole.

Recurrent neural network serves as the framework of encoder to decoder. When predicting trajectory data, RNN network is used as encoder and the hidden layer of the last time dimension is used as output. We mark it as C, then decode C, and finally calculate Y and classify it.

Attention mechanism is similar to traditional encoder-decoder model. But instead of calculating y indiscriminately, the effect of input on output at different times is treated differently.

In the encoder-decoder structure, Encoder encodes all the input sequences into a unified semantic feature C and then decodes it. Therefore, C must contain all the information in the original sequence, and its length becomes the bottleneck that limits the performance of the model. Such as human mobility prediction problems, when the trajectories sequence to be predicted is too long, a C may not store so much information, which will cause a decline in accuracy. Therefore, researches will use multiple C as key.

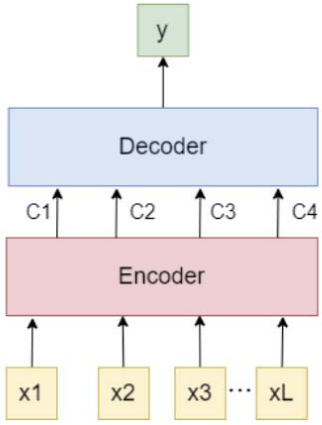


Figure 5: attention sequence to sequence

$$y = f(C1, C2 \ldots CL) \tag{7}$$

There is an alignment mechanism between the input end and the output end, and the input end itself can construct an attention mechanism and learn global word information across distance, which is the self-attention mechanism. Self-attention mechanism can learn contextual information just like RNN, and has the ability of long-distance learning

2.2 Transfer learning

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

2.2.1 Measure criterion

Measurement is not only a basic tool used in disciplines such as machine learning and

statistics, but also an important tool in transfer learning. Its core is to measure the difference between two data domains. Calculating the distance and similarity between two vectors (points, matrices) is the basis of many machine learning algorithms, sometimes a good distance measure can determine the final result of the algorithm is good or bad.

KL divergence and JS distance

KL divergence and JS distance are widely used measures in transfer learning. The Kullback–Leibler divergence, also called relative entropy, is a measure of how one probability distribution P(x) is different from a second, reference probability distribution Q(x).

$$D_{KL}(P||Q) = \sum_{i=1} P(x) log \frac{P(x)}{Q(x)} \tag{8}$$

In probability theory and statistics, the Jensen–Shannon divergence is a method of measuring the similarity between two probability distributions and it comes from KL divergence.

$$JSD(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$$
$$M = \frac{1}{2}(P + Q) \tag{9}$$

Maximum Mean difference (MMD) is the measure with the highest frequency of use in transfer learning. MMD measures the distance between two distributions in the reproduced-kernel Hilbert Space (RKHS) [43] and it is one kernel learning method. For two sets of random variables with n1 and n2 elements respectively, the MMD distance between the two random variables is

$$MMD^2(X, Y) = \left\| \sum_{i=1}^{n_1} \emptyset(x_i) - \sum_{i=1}^{n_2} \emptyset(y_i) \right\|_H^2 \tag{10}$$

Where $\emptyset(\cdot)$ is the mapping, which is used to map the primitive variable to the regenerative kernel Hilbert space.

2.2.2 Model pre-training transfer method

If you already have a model f_s that has been trained on the source domain, and the target domain itself has some tagged data to learn from, you can apply f_s directly to the target domain with some fine-tuning. At this point, you can focus on what happens to the target domain during the fine-tuning process without extra consideration for the migration regularization terms (or all of that). This pretrain-finetune model has been widely used in computer vision (such as pre-trained models on ImageNet), natural language processing (Transformer, Bert) and other fields.

This approach assumes that data in the source domain can share some parameters of the model with data in the target domain. This approach usually works especially well when we have a model trained on a large dataset and we want to perform the same tasks on a small dataset. However, the effectiveness of this approach depends largely on the differences between the two data sets.

Fine-tune could greatly decrease cost of training and expands our data set to some extent.

2.2.3 Domain Adaptation

Domain adaptation is a field associated with machine learning and transfer learning and it is a field that aims at mapping the data distributed in different source and target domains to a feature space to make it as close as possible in that space.

Firstly, it is necessary to introduction some concepts in domain adaptation and transfer learning.

Domain is object of learning and source of knowledge. The domain consists mainly of two parts, one is the data, and the other is the probability distribution that generates the data. In general, D is used to refer to a domain, and P is used to refer to a probability distribution. The domain adaptive problem is designed with two concepts, one is the source domain and the other is the target domain. Domain adaptation takes the knowledge

learned from the source domain with annotated data and applies it to the target domain without annotated data.

Tasks are the objectives of model learning, such as classification tasks and regression tasks. A task usually consists of two parts, a label and a function corresponding to the label.

In most cases, merely the source domain has annotated data and the target domain just has training datasets. Because there's no label data in target domain, it’s impossible to train a model just using target domain’s data by deep learning method.

Virtually, some other studies attempt to migrate a model with low data quality and low performance to a model with high data quality and high performance. In thesis, the domain of zdc data is source domain, which has labels and high-quality, dense data and its model has a relatively good effect. Compared with it, the domain of foursquare data is target domain, which has sparse data and its model is difficult to have a great result.

Domain adaptation could be divided into Discrepancy-based methods, Adversarial-based methods and Reconstruction-based methods. This article will explore only two of the methods used, Discrepancy-based method and Adversarial-based methods.

Deep Domain Confusion (DDC)

By adding an adaptive layer between the source domain and the target domain and adding a domain confusion loss function, the network learns how to classify and at the same time reduces the distribution difference between the source domain and the target domain, so as to realize the domain adaptation.

The network consists of two flows. The first flow is labeled with the input of source data. The input of the other stream is the target data, which contains a small amount of labeled data or no labeled data. The two streams of Convolutional Neural Network share weights. Different from the past, the author adds an adaptation layer between the feature layers of the two flow networks, and calculates a domain loss through the output of the adaptation

layer. In this paper, the MMD (Maximum Mean discrepancies) distance between the features of the source domain and the target domain is used as domain loss, and the difference between the source domain and the target domain is reduced by minimizing the MMD distance.

In the selection of the position of the adaptive layer, the MMD distance between the source data and the target data is calculated layer by layer, and the position of the layer with the minimum MMD distance is selected as the position of the adaptive layer. After the position of the adaptation layer is determined, the adaptation layer of different sizes is also tried, and the size that can minimize the MMD distance is selected as the size of the adaptation layer.

Domain-Adversarial Training of Neural Networks (DANN) [41] is a representative method of domain adaptive problem research based on antagonistic thought.

DANN's network can be divided into three parts, including feature extractor G_f, label predictor G_y and Domain classifier G_d.

Among them, the function of feature extractor is to extract data features required by subsequent tasks on the one hand, and on the other hand, to generate features that cannot be correctly classified by domain classification, so as to realize the alignment of the distribution of feature space between source domain samples and target domain samples; Class classifiers are used to classify sample features; The domain classifier is used to determine whether the feature information extracted by the feature extractor comes from the source domain or the target domain.

The basic principle of DANN is to draw on the idea of GAN. The feature extractor corresponds to the generative antagonistic network in GAN, and the domain discriminator corresponds to the discriminant network in GAN. The sign of the completion of network training is that the domain discriminator cannot distinguish whether the features generated by the feature extractor come from the source domain data or the target domain data. In this case, the feature space of the source domain and the target domain will be

completely mixed together and become the domain invariant feature space.

In the process of model optimization, the training goal is to make the feature generator generate the same distributed features as far as possible for both the input from the target domain data and the input from the source domain data, while the goal of the domain classifier is to distinguish whether the data features come from the source domain data or the target domain data as far as possible. In order to achieve this goal, the model in the feature extractor and classifier increase the gradient between flip layer (GRL), in the process of back propagation network, gradient becomes the original opposite, will lead to back propagation, the domain of discriminator before GRL layer parameters update direction to move in the direction of the loss function is the minimum, The parameters of the feature extractor after the GRL layer will move to the direction of the increase of the loss function, so as to achieve the effect of confrontation.

The loss function of DANN is:

$$C_0(\theta_f, \theta_y, \theta_d) = \frac{1}{n_s}\sum_{x_i \in D} L_y\left(G_y\left(G_f(x_i)\right), y_i\right) - \frac{\lambda}{n}\sum_{x_i \in (D, UD_i)} L_d\left(G_d\left(G_f(x_i)\right), d_i\right) \quad (11)$$

$n = n_s + n_y$, λ is he parameters and it changes with training. After the model converges, parameters will become parameters that make the model reach the local optimal model.

$$(\theta_f, \theta_y) = argmin_{\theta_f, \theta_y} C_0(\theta_f, \theta_y, \theta_d) \quad (12)$$

$$(\theta_d) = argmax_{\theta_d} C_0(\theta_f, \theta_y, \theta_d) \quad (13)$$

Figure 6: Network of DANN [41]

2.3 Federated learning

2.3.1 Categories

According to the different data distribution of participants, Yang Qiang divided federated learning into three types: horizontal federated learning (HFL), vertical federated learning (VFL) and federated transfer federated learning (TFL).

Horizontal federated learning is introduced in the scenarios that data sets share the same feature space but different in samples (like users). Larger overlap of features of the two data sets. From this study, two datasets have same feature space and different data, therefore I use just horizontal federated learning in this study.

Vertical federated learning is applicable to the cases that two data sets share the same sample ID space but differ in feature space. Larger overlap of same IDs(users) of the two data sets.

Federated Transfer Learning applies to the scenarios that the two data sets differ not only in samples but also in feature space. Merely no overlap between two data sets.

Figure 7: Categorization of federated learning [42]

2.3.2 FedAvg

The FEDAVG algorithm integrates multiple deep learning models using SGD into a global model.

Similar to stand-alone machine learning, the goal of federated learning is also experiential risk minimization. The object function is

$$min_{x\in R^d}[F(x) = \frac{1}{n}\sum_{i=1}^{n} f(x; s_i)] \quad (14)$$

N is the sample size, s_i represents sample i, $f(x, s_i)$and represents the loss function of the model s_i. Suppose there are k local models and P_k represents the ordinal set of sample individuals of model k. Assume $n_k = |P_k|$, the loss function is:

$$F(x) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(x) \quad (15)$$

$$F_k(x) = \frac{1}{n_k}\sum_{i\in P_k} f(x; s_i) \quad (16)$$

It is important to note that since the data of each terminal device does not represent the global data, it means that $E_{p_k}[F_k(x)]$ is different from F(x). That is, any local model cannot be considered the global model.

One Update in local model could be thought as an iteration. Assume b as one batch, the formula of the iteration of local model k is:

$$x_k \leftarrow x_k - \frac{\eta}{|b|}\sum_{i\in b} \nabla f(x_k; s_i) \quad (17)$$

The idea of FEDAVG algorithm is very intuitive. The training process is divided into several rounds. In each round, selected CK($0 \leqslant C \leqslant 1$) models to train. The number of epochs of the local model k in a round is E, batch size is B, so the number of iterations is $\frac{En_k}{B}$. At the end of a round, the global model is obtained by weighted averaging the parameters of all the local models participating in the learning. The detailed algorithm is as follows

Algorithm 1: FedAvg

Input: number of models K, number of epochs E, batch size B, learning rate η

Output: global model parameter x

Initialize x;
for each round t=1, 2, …, **do**
    $n_k \leftarrow |P_k|$;
    $m \leftarrow max(CK, 1)$;
    $S_t \leftarrow random\ set\ of\ m\ local\ models$;
    for each local model in parallel **do**
        $x_k \leftarrow x$
        $\beta \leftarrow split\ P_k\ into\ batches\ of\ size\ B$
        for each epoch i from 1 to E do
            for each batch $b \in \beta$ d
                $x_k \leftarrow x_k - \frac{\eta}{B}\sum_{j \in b} \nabla f(x_k; s_j)$
            end
        end
    end
$x \leftarrow \sum_{k \in S_t} n_k$;
$x \leftarrow \sum_{k \in S_t} \frac{n_k}{n} x_k$;
end
return x;

Chapter 3. Problem statement

In this section, we define the terms and concepts frequently used throughout this thesis.

Definition 3.1 (Raw GPS data). Each reading from a localization sensor can be described as a 3-tuple of the time stamp, latitude and longitude. Thus, raw GPS data collected by the mobile device u can be formally represented as follows:

$$X_u = \{(t, lat, lon)\} \tag{18}$$

Thus, the trajectory of each user raw_traj_u can be defined in the following manner of:

$$raw\_traj_u = x_{u,0}, x_{u,1}, \dots x_{u,n}, \ \ x_{u,i} \in X_u \tag{19}$$

where $x_{u,i}$ is the i-th record (ordered by time) of user u.

Definition 3.2 (Location Cluster). To make it easier for learning a spatial multimodality, we introduce location clusters $\{c_m = (lat_{m,} lon_m) | m = 1, \dots M\}$ (as shown in Figure 3) to transform the geo-coordinates (latitude, longitude) in will to the nearest location cluster.

Definition 3.3 (Cluster-level trajectory). A cluster-level trajectory traj is represented by the sequence of cluster IDs:

$$traj = c_0, c_1, \dots, c_{T-1} \tag{20}$$

where T is the length of the trajectory. Typically, the time interval between two adjacent cluster IDs is 15 min.

Definition 3.4 (Past/most recent trajectory). To model both the long- and short-term dependency respectively, we slice the trajectory of user u into the past trajectories $\{traj_u^d | d \in D_u\}$ on day d and the most Δt recent trajectory $traj_{u,t-\Delta t:t}$. Note that users may leave the studied region or switch off their mobile phone, and thus we do not expect we have user data on every day in our dataset. Thus, we use D_u to denotes the set of days in which the user has records within the studied region.

Definition 3.5 (Human Mobility prediction). In this study, we predict the future Δt-aheadlocation $traj_{u,t+\Delta t}$ based on the most recent Δt observations $traj_{u,t+\Delta t}$ and past trajectories $\{traj_u^d | d \in D_u\}$from the same user.

Thus, we can formulate the prediction task at time t as modeling the conditional probability:

$$p(traj_{u,t+\Delta t}|traj_{u,t-\Delta t}, \{traj_u^d|d \in D_u\}) \quad (21)$$

Note that, Δt is used for both Δt-ahead prediction and Δt most recent observations.

Definition 3.6(domain adaptation)

Given the source language $D_s = \{x_i, y_i\}_{i=1}^n$ and a target domain $D_t = \{x_i\}_{i=n+1}^{n+m}$, when their feature spaces ($X_s = X_t$), category Spaces ($Y_s = Y_t$), and conditional distribution probabilities ($P(y_s|x_s) = P(y_t|x_t)$are the same, the edge distributions of the two domains are different($P(x_s) \neq P(x_t)$). The goal of domain adaptation is to use labeled data D_s to learn a classifier $f: x_t \rightarrow y_t$ to predict the label $y_t \in Y_t$ of the target domain D_t.

Definition 3.7(federated learning)

Define N data owners $\{F_i, \dots F_n\}$, all data owners want to train a machine learning model by merging their respective data $\{D_i, \dots D_n\}$. Centralized method will put all data together to train a model M_{sum}. But a federated learning method will let every data owner F_i collaboratively train a model M_{fed} without expose its data D_i to other data owners. Besides, the accuracy of M_{fed} ,denoted as V_{fed} should be greatly close to the actuary of V_{sum}. Suppose δ as a non-negative real number, if $|V_{fed} - V_{sum}| < \delta$
we say that the federated learning algorithm has δ-accuracy loss.

Chapter 4. Data pre-processing

4.1 Dataset

In this research, I mainly use two datasets. **Foursquare data** (LBSNs sparse data) and **zdc data** (dense trace data).

Foursquare dataset is one kind of Location-based Social Network trajectories data and it collected by web spider. After filtering out noise and invalid check-ins, it contains check-ins about 10 month (from 12 April 2012 to 16 February 2013) and 573,703 check-ins, 2273 active users in Tokyo and each check-in are associated with its time stamp, its GPS coordinates and its semantic meaning (represented by fine-grained venue-categories), such as train station, library and etc. Average time interval of every check-in is 2.44 days. Therefore, it is one kind of greatly sparse human mobility data, which will make it is difficult to predict or build a highly accuracy model.

Zdc dataset is one kind of people flows data, which is collected by individual location data sent from smart phones with enabled AUTO-GPS function under users' consent by "docomo map navi" service provided by NTTDOCOMO, INC. Those data are processed collectively and statistically so that it won't reveal users' private information. Original data just includes GPS data (latitude, longitude), time stamp and user id.It relabeled users' id and doesn't reserve any users' private information such as gender or age. It has millions of users and average time interval is 15 minutes, so it is large-scale and dense trajectories. Virtually, some previous studies have utilized it on trajectory prediction and has a relatively great result. Because the size of complete zdc data that occurs in same period with foursquare is too large, I just use 3 month zdc data and random select 1/10 users in it as part zdc dataset used in experiment.

Virtually, although the two datasets are similar in the shape of data and period, they are greatly different from data accuracy, scale and time interval (shown in Table 1).

Dataset	Foursquare	Zdc
#Users	2273	14140
#Poi	1082	1600
#Trejectories	573,703	millions
Collected period	12/2012-2/2013	12/2012-2/2013
Time Interval	2.44 days	15minutes

Table 1: Statistics of the datasets

4.2 Data pretreatment

Through I have introduced condition of foursquare and zdc dataset in Tokyo, virtually original data is collected in larger scale. Considering latter experiment, it is necessary to make the two datasets in same region. I find out the border of Tokyo and remove data not in this area for the two datasets.

In addition, as is mentioned before (shown in Table 2), the types of the two datasets are greatly different. Except time interval and data scale, Foursquare data is check-in data that adjacent location is different in most times. Zdc data is dense traces, within 15 minutes, user is probably in same location in many records. Therefore, it is necessary to make the two datasets more closely for data fusion. If one user stays in one region over a given time, we could consider this region is a stay point for this user. Fortunately, scikit-mobility has this function and I utilized it to preprocess zdc data as the form of stay point, making the data shape more closely to foursquare data. After this step, adjacent location will be different for every user. From this angle, two datasets become closer, which will make it easier fuse two datasets.

Besides, I also rename every user id for two datasets. Although there is possibly some overlap between the users of the two datasets, I make the user numbers of the two datasets completely different in order to avoid confusion in model training. I also completed some pre-processing on time and the unit of time is seconds, which make sure that two datasets have uniform format.

4.3 trajectory cluster

The purpose of this thesis is predicting user's next location. Ideally, we should predict user's next accurate location, such as latitude and longitude. Virtually, this kind of continuous GPS data is difficult to learn even we use a complex model because a complex transportation network makes the spatial dependency highly non-linear. Another advantage of discrete value in human mobility data is that it is more suitable to characterize spatial multimodal distributions. Therefore, it is necessary to convert continuous GPS data to discrete value.

Although foursquare data already has location id feature, we don't know how it is recorded and to fuse it with zdc data, label the two datasets by same method is necessary. In this research, I utilize grid approximately substitute for accurate location.

I split Tokyo into 1km×1km grids in a given latitude and longitude, then count numbers of trajectories in every grid. Considering the data size of zdc data is greatly bigger than foursquare data, I just use zdc data in this data cluster. As is well known that the more classes, the harder the model is to learn, it's impossible to use all grids. So, I chose 1600 most common anchors in Tokyo. Calculate the distance between each piece of data and each anchor, then select the anchor that is closest to each piece of data as location label for each piece of data. As same as treatment of zdc data, I use same method to relabel foursquare data. Table 3 shows statistical information of this data cluster.

	Min(km)	Max(km)	Median(km)	Mean(km)
Zdc data	0.92	192.42	0.92	1.32

Table 2. Adjacent Cluster Center Distance Statistics [45]

By these anchors (shown in Figure3), the two datasets have same form labels. As is mentioned before, I chose 1600 most common anchors in Tokyo by zdc data. However, foursquare data doesn't have all the anchors, it is concentrated in 1082 anchors, which means that foursquare data appears in fewer anchors in Tokyo.

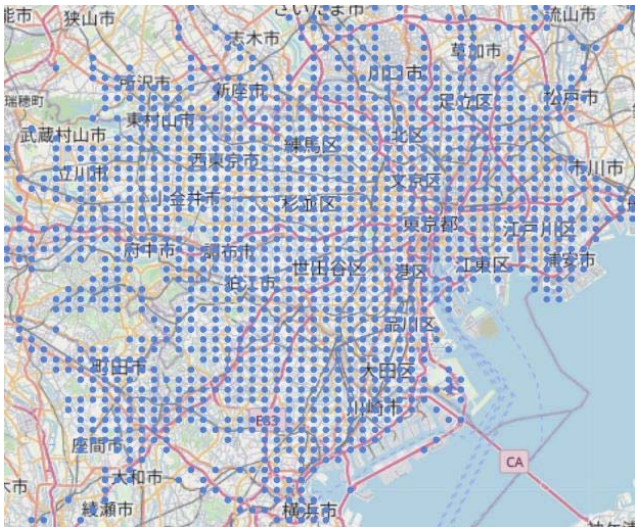


Figure 8:1600 common anchors in Tokyo

Chapter 5. Model

5.1 Activation function

Activation Function is a function added to an artificial neural network to help the network learn complex patterns in data. Similar to the neuron-based model of the human brain, the activation function ultimately determines what to fire to the next neuron.

Sigmoid function

The Sigmoid function outputs from 0 to 1. Since the output values are limited to 0 to 1, it normalizes the output of each neuron. It is well suited to models that use predicted probabilities as outputs.

$$f(z) = \frac{1}{1+e^{-z}} \tag{22}$$

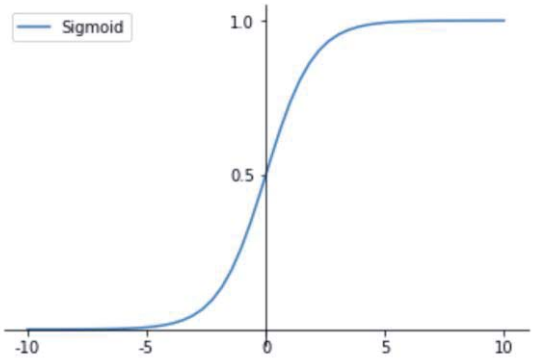


Figure 9: Sigmoid function

Tanh function

In a general binary classification problem, the tanh function is used for the hidden layer, while the sigmoid function is used for the output layer

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{23}$$

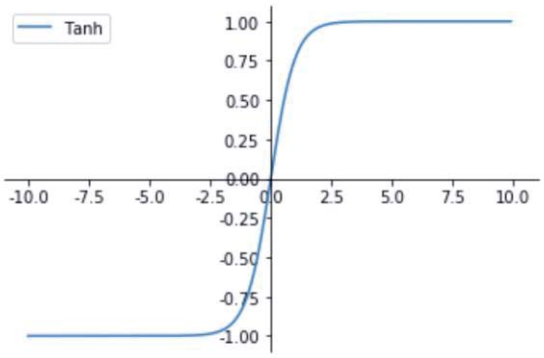


Figure 10: Tanh function

ReLU function

Since there is only a linear relationship, the calculation speed of ReLU function is relatively fast, so it is widely used in deep learning

$$\sigma(x) = \begin{cases} max(0, x) & , x \geq 0 \\ 0 & , x < 0 \end{cases} \tag{24}$$

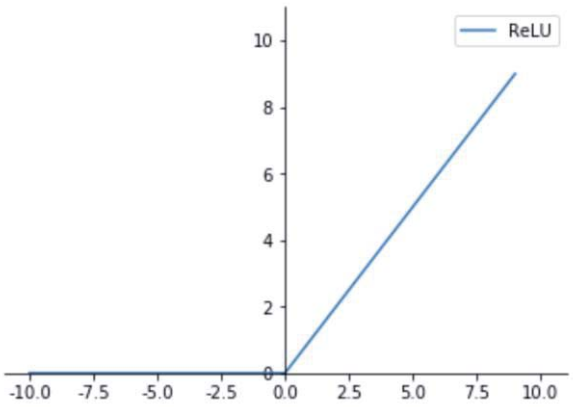


Figure 11: Relu function

SoftMax Function

SoftMax is an activation function for multi-class classification problems where more than two class tags require class membership. For any real vector of length K, SoftMax can compress it into a real vector of length K, with a value in the range (0,1), and the sum of the elements in the vector is 1.

$$f(x) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{25}$$

5.2 Baseline

This research attempts to predict long-term and sparse human mobility pattern. Therefore, it is practically impossible to have a great result if we just use RNN, even LSTM. Fortunately, there have some researches centralized in modeling sparse human mobility data and one of them is Flashback [24].

Flashback utilizes spatiotemporal contexts to predict sparse data. Figure 3 shows mechanism of it. It is designed for modeling sparse user mobility trajectories Compared with vanilla RNN, it adds attention mechanism and user embedding to consider personalization. Instead, implicitly add spatial-temporal factors into the recurrent hidden sate in RNN model, it uses attention mechanism to search past hidden state. Then the hidden state that has high predictive power will have higher weight and the hidden state that has lower predictive power will have little influence on the final output. Therefore, this model will learn greatly from rich spatiotemporal contexts in past mobility trajectories.

It computes the average weight of the historical hidden state h_j and the weight is calculated by the temporal and spatial distance($\Delta T_{i,j}$ and $\Delta D_{i,j}$), which could represent spatiotemporal contexts between adjacent location in trajectories sequence.

From a temporal perspective, we need to consider the influence of time. Considering human mobility pattern changes periodically because human behavior should follow the same pattern every day or week or over a certain time period, it uses a Havercosine function, which is a typical periodic function and outputs bounder form 0~1, as follows:

$$\omega_{period}(\Delta T_{i,j}) = hvc(2\pi\Delta T_{i,j}) \tag{26}$$

$where\ hvc(x) = \frac{1+cos(x)}{2}$

Virtually, the influence of time changes with exponentially. As is well know that old trajectories will has less influence on prediction task. To model this factor, this framework uses a temporal exponential decay.

$$\begin{aligned}\omega_T(\Delta T_{i,j}) &= \omega_{period}(\Delta T_{i,j}) \cdot e^{-\alpha \Delta T_{i,j}} \\ &= hvc(2\pi \Delta T_{i,j}) \cdot e^{-\alpha \Delta T_{i,j}}\end{aligned} \quad (27)$$

Where α could control how fast the weight decrease by time $\Delta T_{i,j}$ as a temporal decay rate.

From a spatial perspective, we also need to consider the influence of distance between adjacent location. If the location is closer to the current location, it will be more helpful for location prediction. Similar to time interval, it also models as a exponential way as:

$$\omega_S(\Delta D_{i,j}) = e^{-\beta \Delta D_{i,j}} \quad (28)$$

Where $\Delta D_{i,j}$ is the L2 distance (Euclidean distance) between two POIs p_i and p_j, and β controls how fast the weight decreases by spatial distance $\Delta D_{i,j}$ as a spatial decay rate. It is necessary to combine these two weights separately calculated by spatial and temporal factors.

$$\begin{aligned}\omega_T(\Delta T_{i,j}, \Delta D_{i,j}) &= \omega_T(\Delta T_{i,j}) \cdot \omega_S(\Delta D_{i,j}) \\ \omega_T(\Delta T_{i,j}, \Delta D_{i,j}) &= hvc(2\pi \Delta T_{i,j}) \cdot e^{-\alpha \Delta T_{i,j}} \cdot e^{-\beta \Delta D_{i,j}}\end{aligned} \quad (29)$$

In formula 29, it both consider the effects of time and space and it will be multiplied by hidden state in order to consider spatial-temporal context.

In addition, this model also uses embedding vector for every user. Then it aggregated hidden state with user embedding vector in to a fully connected layer in order to predict the next location

Compared with previous study, this framework has already a relatively great result when predicting sparse traces.

Figure 12: architecture of Flashback [24]

5.3 Federated learning model

Although traditional federated learning methods usually have many parts, especially when it is used in mobile devices, there are only two parts, foursquare dataset and zdc dataset, in this research.

Instead training models by two datasets separately, I train two local models of two datasets and aggregate them as global model by FedAvg. Afterwards, use global model to update the two local models and continue the iteration.

In this model, two local trainer uses baseline network Flashback. Therefore, it could be considered as average performance of two datasets to some extent.

The detailed procedure is as follows:
Step1: Split two datasets by batch size
Step2: Initialize parameters of two local models and global model
Step3: for every epoch
for every local model
train local model using Flashback by batch size
update local model by gradient descent
update global model by averaging parameters of local models
update local models by global model
end
end

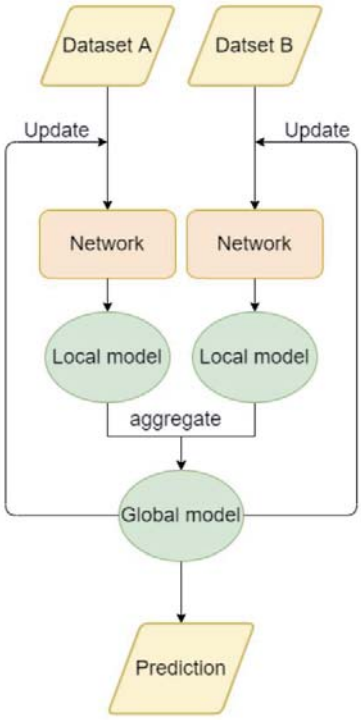


Figure 13: Federated learning process

5.4 Transfer learning model

Based on this framework, I design two transfer learning model. One is based on Deep Domain Confusion (DDC), I calculate mmd loss when training [44]. Another is based on DANN, except classifier, I add discriminator to it to predict which domain the data from.

Model based on DDC calculated mmd loss before fully connected layer between two models trained by foursquare dataset and zdc dataset.

The loss function will become:

$$L(X_t) = \frac{1}{n}\sum_{x_i \in X_t} F(x_i) + \lambda \cdot \left(\sum_{h_i \in H_t, h_j \in H_s} MMD(h_i, h_j)\right) \quad (30)$$

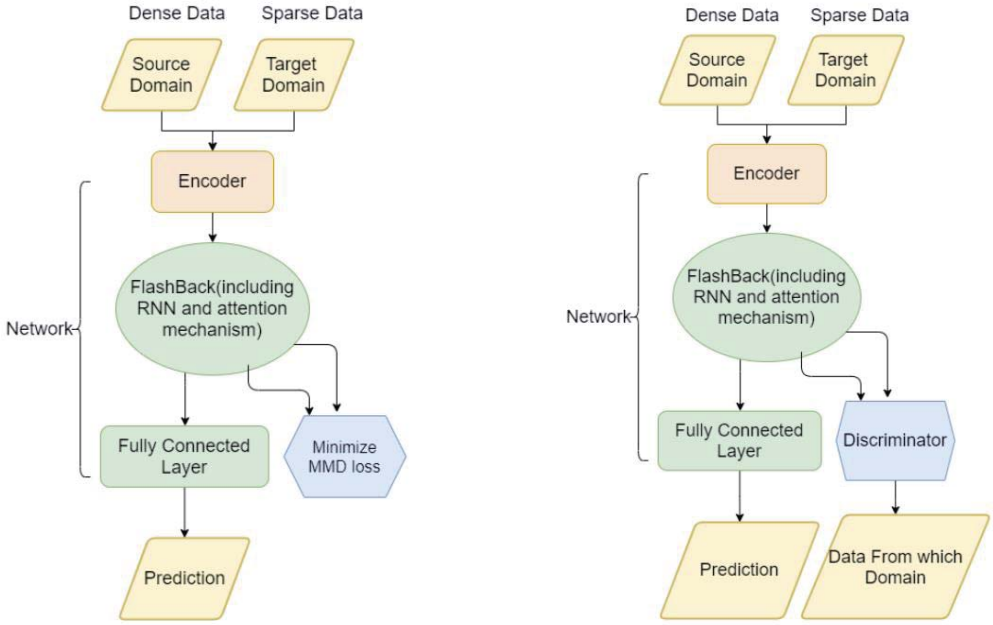
Where t means data from target domain and s means data from source domain is cross entropy that obtained by basic model Flashback, λ is transfer parameter that controls the influence of transfer learning. MMD calculates mmd loss of two datasets and input of G

is vector of hidden states(h)

$$L(X_t) = \frac{1}{n}\sum_{x_i \in X_t} F(x_i) + \lambda \cdot \left(\sum_{h_i \in H_t, h_j \in H_s} D\left(h_i, h_j\right)\right) \tag{31}$$

Where t means data from target domain and s means data from source domain is cross entropy that obtained by basic model Flashback, λ is transfer parameter that controls the influence of transfer learning. D calculates domain classifier loss of two datasets and input of G is vector of hidden states(h), which means D calculates the input origins from which domain.

These two models both used domain adaptation to align two kinds of data to have a better result.



Discrepancy-based method Adversarial-based method

Figure 14: Transfer learning model

5.5 Federated transfer learning model

I used both federated learning and transfer learning in the final model. It adapts the inputs from source domain by adding an alignment layer. Both of the previous two transfer

learning methods directly input data into the model during training. However, in consideration of privacy protection, direct interaction between data cannot be conducted. Therefore, the federated transfer learning model eventually used is different from methods that I used before, including DCC and DANN.

Given the network from server and two users, I add a correlation alignment layer after fully connected layer in order to adapt the domains. This alignment function could be used to align the second-order statistics between the inputs. The loss of correlation alignment can be calculated as follows:

$$l_{coral} = \frac{1}{4d^2}||C_s - C_t||_F^2 \tag{32}$$

where $||\cdot||_F^2$ is squared matrix Frobenius norm, and d demonstrates the dimension of embedding features. C_s and C_t denotes the covariance matrices of the source(global) domain and target(local) domain's weights. Compared with training with both regression, CORAL loss could learn features that could perform greatly on target domain(local).

Therefore, the loss function of this method is:

$$arg\ min\ l_u = \sum_{i=1}^{n_u} l({y_i}^u, f_u({x_i}^u)) + \eta L_{coral} \tag{33}$$

Where λ demonstrates the trade-off parameter.

Figures 15 shows federated transfer learning model I proposed in this study.
The detailed procedure is as follows:
Step1: Split two datasets by batch size
Step2: Initialize parameters of two local models and global model
Step3: for every epoch
for every local model
train local model using Flashback by batch size
minimize correlation alignment loss between local model and global model
update local model by gradient descent
update global model by averaging parameters of local models
update local models by global model
end
end

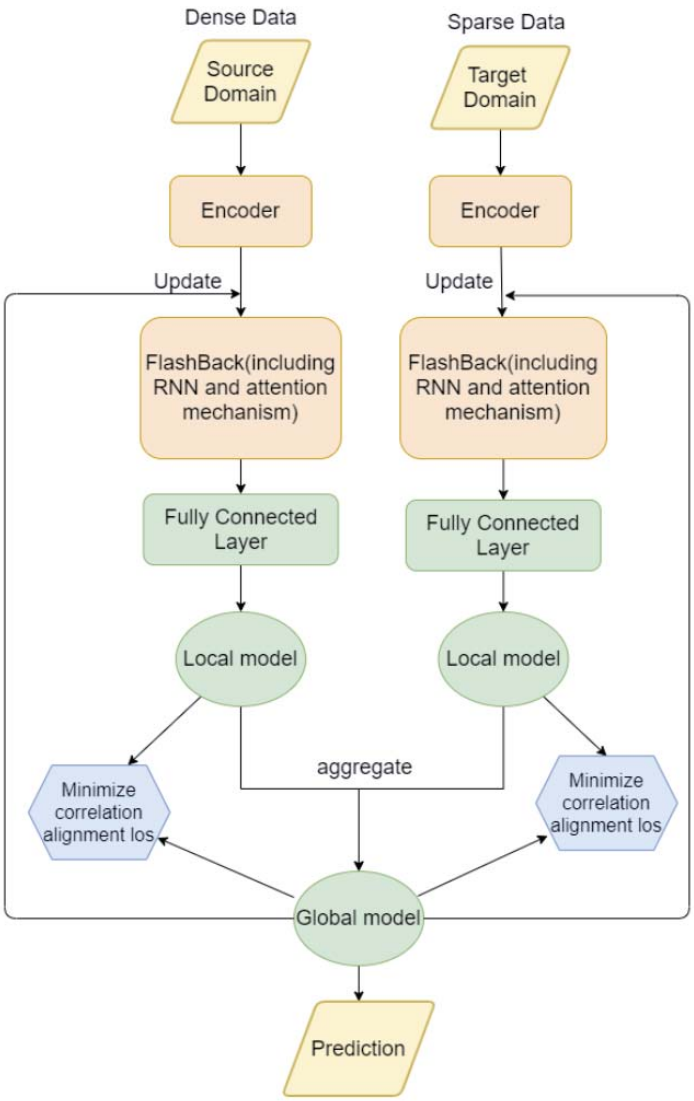


Figure 15: Federated transfer learning model

Chapter 6. Experiment and analysis

6.1 Measure criterion

Cross entropy

In information theory, the cross-entropy between two probability distributions p and q over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set if a coding scheme used for the set is optimized for an estimated probability distribution q, rather than the true distribution p.

The formula 34 for cross entropy is as follows

$$H(p,q) = -\sum_{i=1}^{n} p(x_i) log(q(x_i)) \quad (34)$$

In machine learning, we need to evaluate the gap between the label and predicts, using KL divergence is just right. Due to the KL divergence in the first part H (y) is changeless, therefore in the process of optimization, the need to focus only on the cross entropy. Thus, cross entropy is generally directly used to make loss evaluation model in machine learning

Precision

Precision measures how accurate is your predictions, the percentage of your predictions are correct.

$$precision = \frac{TP}{TP+FP} \quad (35)$$

where TP is true positive, FP is false positive.

Recall

Recall measures how good you find all the positives. For example, we can find 80% of the possible positive cases in our top K predictions.

$$recall = \frac{TP}{TP+FN} \quad (36)$$

where FN is false negative

recall@10 means probability that we find positives from top10 data

mAP

The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above (usually x axis is recall, y axis is precision).

$$\mathrm{AP} = \int_0^1 \mathrm{p(r)d_r} \tag{37}$$

mAP is mean Average Precision

Because we have many batches, so I will calculate average ap for evert batch. That is mAP

6.2 Experiment setups

The experiments are conducted on a Linux(ubuntu) sever with Intel Xeon E5-2690v4 CPU (2.6GHz 14C 35M 9.60 GT/sec 135W), 2x TitanX Pascal 12GB GDDR5X graphics card, 128GB (8x 16GB DDR4-2400 ECC RDIMM) and a 1.2TB Intel NVmeDC P3600 Series SSD. I use 80% of the dataset as training data and 20% for testing data.

In order to compare with the results of different experiments, I use same parameter in most experiment. The length of input trajectories sequence is 20, the batch size is 1024(means 1024 users in a batch), length of hidden state is 10, initial learning rate is 0.01, number of locations is 1600 and transfer loss parameter in transfer learning is 10.

As is mentioned in chapter 3, original foursquare dataset already has labels. Therefore, firstly I conduct the experiment on original foursquare dataset and relabeled foursquare datasets by basic model Flashback. After that, I also use zdc dataset to complete same experiment.

Afterwards, I conduct the experiment of federated learning methods on foursquare dataset and zdc dataset. I use method of FedAvg to aggregate two local model as global model except user embedding layer.

Then, two transfer learning models are used to predict next location of trajectories in

foursquare dataset by the help of zdc data. Because we want to improve model performance of foursquare dataset, zdc data is source domain and foursquare data is target domain.

Finally, I use federated transfer learning model that I proposed in chapter 4 to predict human mobility data without privacy disclosure. Significantly, different from just transfer learning method, in this experiment, source domain is weight of global model and target domain is weight of local model, including local trainer of foursquare data and zdc data.

6.3 Result of location prediction

Firstly, I use basic model flashback on zdc and foursquare dataset separately. As is shown in figure, the loss of model trained by zdc is greatly smaller than that of foursquare dataset. Of course, except loss, accuracy of model of zdc data is also better than foursquare data because zdc data is denser and has relatively high-quality data, which means the trajectories contain more information. It should be noted that foursquare dataset has same kind of label with zdc data in this controlled trial.

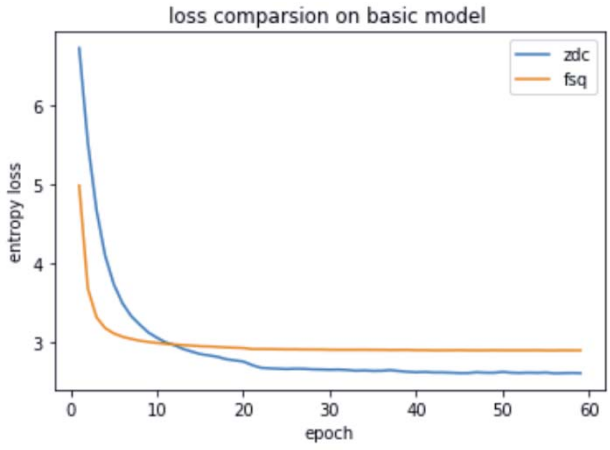


Figure 16: loss comparison on basic model

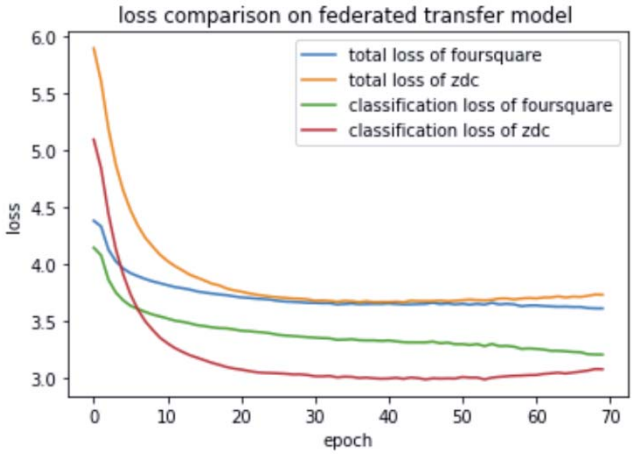


Figure 17: loss comparison on federated transfer learning model

Different from performance of basic model, after we import transfer learning and federated learning, the total loss of two local models that trained by local data and updated by global model become closer, which means that two datasets are aligned to some extent. It seems that the total loss is in Figure 17 larger than loss in Figure 16 because the calculation methods of loss are different. The loss in Figure 16 is classification error and the total loss in figure 17 is computed by classification loss and transfer loss. Virtually, if we just compare classification loss, the classification loss in Figure 17 is smaller than that in Figure 16.

Data	mAP	Recall@10	Model
Original fsq data	48.354088%	75.302128%	Flashback
Relabeled fsq data	48.219626%	75.302128%	Flashback
Zdc data(part)	53.130656%	77.117857%	Flashback
Relabeled fsq data	47.98230%	74.808568%	FedAvg method
Relabeled fsq data	48.710939%	75.575331%	DDC transfer method
Relabeled fsq data	48.672451%	75.680877%	DANN transfer method
Relabeled fsq data	**48.416047%**	**75.509106%**	**Federated transfer method**

Table 3: Location prediction performance on different datasets with different models

6.4 Analysis

From result of location prediction, we could know that basic model trained by zdc data has better performance due to its density and rich spatial-temporal information than basic model trained by foursquare data, which makes transfer learning become meaningful.

Caused by distribution divergence between foursquare dataset and zdc data, it's difficult to have a good performance if we just combine these two datasets even in centralized method. Thus, when we just use federated learning on basic model, the performance of model actually declines because global model couldn't balance prediction task of two datasets with great distribution divergence.

Just as we suspected before, the performance of model trained by foursquare data improves when we use transfer learning model. By domain adaptation, we could transfer source domain (zdc data) to target domain (foursquare data) to some extent. Because zdc data is more high-quality, when we align the distribution of two datasets, the model could learn some extra spatial-temporal information from zdc data. Benefit from it, model trained by foursquare data could make up missing spatial-temporal information due to sparsity of data.

Finally, when we use federated transfer learning model, the performance of model is still better than basic model. Unavoidably, federated learning method will decrease model accuracy to some extent because domain adaptation is not directly done between two datasets. Virtually, the alignment of two local models trained by two datasets is completed by global model indirectly.

There are some possible reasons that cause improvement of model is not so great. One is the complexity of the task because of classes of this model is 1600. As is well known that the more classes of task, the difficult the model to learn. Another reason is that the periodicity of zdc data I used in this experiment is not long enough. The time span of foursquare data is over one year, but I just use 3 months zdc data because the size of one year zdc is so large, which makes it's hard for zdc data to supplement whole spatial-

temporal information.

Chapter 7. Conclusion and future work

Although it's difficult to predict sparse human mobility data by traditional method, such as RNN and LSTM, model based on attention mechanism and leverage rich spatiotemporal contexts makes it have a relatively high accuracy in prediction. Basic model Flashback uses attention mechanism to search past hidden state that has high predictive power so that it could benefit from spatial-temporal contexts in GPS data.

Based on Flashback, I design two transfer learning networks that could adapt the domain from dense data to sparse data to help improve performance of prediction of sparse human mobility data by domain adaptation in transfer learning. One network is based on DDC and another is based on DANN. Model based on DDC adds a domain adaptation layer that calculates mmd loss between intermediate result, hidden state and minimize this kind of transfer loss so that the source domain could convert to target domain. Model based on DANN adds a discriminator that classifies the input data from which domain by adversarial ideas generated from GAN. Compared with basic model, the two transfer learning models both have better performance on sparse human mobility data prediction.

In order to protect privacy of companies, I utilize federated learning method that could help companies collaborate to train a collective model that they can store their raw data locally and needn't to exchanged data with each other. Virtually, I train two local models by foursquare data and zdc data separately and aggregate them as a global model except user embedding layer. Due to great difference between two datasets, if we only use federated learning, it's difficult to improve performance of prediction of sparse human mobility data. Experiment shows it has a little decline of accuracy.

Combined with the two technologies, this study proposes a federated transfer learning model for modeling sparse human mobility data without exchange of raw data. It is different from transfer learning method that the global model could be considered as source domain and the local model could be considered as target domain. I add a domain

adaptation layer after fully connect layer of network. When training in federated method, the local model will be aligned to global model and the global model is generated by two datasets, which means that the two datasets will be aligned with each other. Experiment shows that the performance of this method will slightly worse than directly using transfer learning method and greater than basic model Flashback.

Although this method could improve performance of prediction of sparse human mobility data, the degree of improvement is little small. Therefore, I will attempt to using other domain adaptation methods to improve the prediction model better in the future. On the other hand, this study just considers two companies' cooperation and virtually, if more companies could attend generate model by federated learning method, the result could be better. I will consider let more parties attend federated learning and make the prediction become better in the future.

Reference

[1] Jakub KoneÄĐnÃ¡, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. In NIPS Workshop on Private Multi-Party Machine Learning.

[2] Pan S J, Yang Q. A survey on transfer learning[J]. IEEE Transactions on knowledge and data engineering, 2009, 22(10): 1345-1359.

[3] Ben-David S, Blitzer J, Crammer K, et al. A theory of learning from different domains[J]. Machine learning, 2010, 79(1): 151-175.

[4] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.

[5] Longbiao Chen, Dingqi Yang, Daqing Zhang, Cheng Wang, Jonathan Li, and Thi-Mai-Trang Nguyen. 2018. Deep mobile traffic forecast and complementary base station clustering for C-RAN optimization. Journal of Network and Computer Applications 121 (2018), 59 – 69.

[6] W Kang Z Li FY Wang Y Lv, Y Duan. 2015. Traffic Flow Prediction with Big Data: A Deep Learning Approach. IEEE Transactions on Intelligent Transportation Systems, 2015, 16(2):865-873 (2015).

[7] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 961–971.

[8] Xuan Song, Xiaowei Shao, Huijing Zhao, Jinshi Cui, Ryosuke Shibasaki, and Hongbin Zha. 2010. An online approach: Learning-semanticscene-by-tracking and tracking-by-learning-semantic-scene. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 739–746.

[9] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Li Zhenhui. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. In The Thirty-Second AAAI Conference on Artificial Intelligence.

[10] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In Thirty-First AAAI Conference on Artificial Intelligence.

[11] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. Mining user mobility features for next place prediction in location-based services. In ICDM, pages 1038–1043. IEEE, 2012.

[12] Dingqi Yang, Bin Li, and Philippe Cudr´e Mauroux. Poisketch: Semantic place labeling over user activity streams. In IJCAI, pages 2697–2703, 2016.

[13] Min Xie, Hongzhi Yin, Hao Wang, Fan jiang Xu, Weitong Chen, and Sen Wang. Learning graph based poi embedding for location-based recommendation. In CIKM, pages 15–24. ACM, 2016.

[14] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T Campbell. 2011. Nextplace: a spatio-temporal prediction framework for pervasive systems. In International Conference on Pervasive Computing. Springer, 152–169.

[15] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. 2015. Regularity and conformity: Location prediction using heterogeneous mobility data. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 1275–1284.

[16] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and Mobility: User Movement in Location-based Social Networks. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’11). ACM, New York, NY, USA, 1082–1090.

[17] Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. Geo topic model: joint modeling of user’s activity area and interests for location recommendation. In WSDM, pages 375–384. ACM, 2013.

[18] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden markov models. In UbiComp, pages 911–918. ACM, 2012.

[19] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In IJCAI, 2013.

[20] Tom´aˇs Mikolov, Martin Karafi´at, Luk´aˇs Burget, Jan ˇCernock`y, and Sanjeev Khudanpur. Recurrent neural network-based language model. In INTER-SPEECH, 2010.

[21] Daniel Neil, Michael Pfeiffer, and ShihChii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In NIPS, pages 3882–3890, 2016.

[22] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In AAAI, 2016.

[23] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. Where to go next: A spatio-temporal gated network for next poi recommendation. In AAAI, volume 33, pages 5877–5884, 2019.

[24] Yang D, Fankhauser B, Rosso P, et al. Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States! [C]//Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. 2020: 2184-2190.

[25] Zhuo H, Yang Q, Hu D H, et al. Transferring knowledge from another domain for learning action models[C]//Pacific Rim International Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 2008: 1110-1115.

[26] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Scholkopf, and Alex J ¨ Smola. Correcting sample selection bias by unlabeled data. In Advances in neural information processing systems, pages 601–608, 2007.

[27] Pipei Huang, Gang Wang, and Shiyin Qin. Boosting for transfer learning from multiple data sources. Pattern Recognition Letters, 33(5):568–579, 2012

[28] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S Yu. Visual domain adaptation with manifold embedded distribution alignment. In 2018 ACM Multimedia Conference on Multimedia Conference, pages 402–410. ACM, 2018.

[29] Sinno Jialin Pan, Ivor W Tsang, James TKwok, and Qiang Yang. Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks, 22(2):199–210, 2011

[30] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Interntional Conference on Machine Learning (ICML), 2015.

[31] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. arXiv preprint arXiv:1412.3474, 2014.

[32] Konečný J, McMahan H B, Ramage D, et al. Federated optimization: Distributed machine learning for on-device intelligence[J]. arXiv preprint arXiv:1610.02527, 2016

[33] EU. 2016. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (general data protection regulation).

[34] Cynthia Dwork. 2008. Differential privacy: A survey of results. In In Theory and Applications of Models of Computation. Springer, 1–19.

[35] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In Advances in Neural Information Processing Systems, pages 4424–4434, 2017.

[36] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning for recommendation. arXiv preprint arXiv:1802.07876, 2018.

[37] Li J, Khodak M, Caldas S, et al. Differentially private meta-learning[J]. arXiv preprint arXiv:1909.05830, 2019.

[38] Li X, Huang K, Yang W, et al. On the convergence of fedavg on non-iid data[J]. arXiv preprint arXiv:1907.02189, 2019.

[39] Malinovskiy G, Kovalev D, Gasanov E, et al. From Local SGD to Local Fixed-Point Methods for Federated Learning[C]//International Conference on Machine Learning. PMLR, 2020: 6692-6701.

[40] Chen Y, Qin X, Wang J, et al. Fedhealth: A federated transfer learning framework for

wearable healthcare[J]. IEEE Intelligent Systems, 2020, 35(4): 83-93.

[41] Ganin Y, Ustinova E, Ajakan H, et al. Domain-adversarial training of neural networks[J]. The journal of machine learning research, 2016, 17(1): 2096-2030.

[42] Yang Q, Liu Y, Chen T, et al. Federated machine learning: Concept and applications[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2019, 10(2): 1-19.

[43] Borgwardt K M, Gretton A, Rasch M J, et al. Integrating structured biological data by kernel maximum mean discrepancy[J]. Bioinformatics, 2006, 22(14): e49-e57.

[44] Tzeng E, Hoffman J, Zhang N, et al. Deep domain confusion: Maximizing for domain invariance[J]. arXiv preprint arXiv:1412.3474, 2014.

[45] Fan Z, Song X, Jiang R, et al. Decentralized attention-based personalized human mobility prediction[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2019, 3(4): 1-26.