DeepStore: An Interaction-aware Wide&Deep Model for Store Site Recommendation with Attentional Spatial Embeddings

Yan Liu, Bin Guo, Senior Member, IEEE, Nuo Li, Jing Zhang, Jingmin Chen, Daqing Zhang, Fellow, IEEE, Yinxiao Liu, Zhiwen Yu, Senior Member, IEEE, Sizhe Zhang, and Lina Yao, Member, IEEE

Abstract—Store site recommendation is one of essential business services in smart cities for brick-and-mortar enterprises. In recent years, the proliferation of multi-source data in cities has fostered unprecedented opportunities to the data-driven store site recommendation, which aims at leveraging large-scale user-generated data to analyze and mine users’ preferences for identifying the optimal location for a new store. However, most works in store site recommendation pay more attention to a single data source which lacks some significant data (e.g., consumption data, user profile data). In this paper, we aim to study the store site recommendation in a fine-grained manner. Specifically, we predict the consumption level of different users at the store based on multi-source data, which can not only help the store placement but also benefit analyzing customer behavior in the store at different time periods. To solve this problem, we design a novel model based on the deep neural network, named DeepStore, which learns low- and high-order feature interactions explicitly and implicitly from dense and sparse features simultaneously. In particular, DeepStore incorporates three modules: the cross network, the deep network and the linear component. In addition, to learn the latent feature representation from multi-source data, we propose two embedding methods for different types of data: the filed embedding and attention-based spatial embedding. Extensive experiments are conducted on a real-world dataset including store data, user data, and POI data, the results demonstrate that DeepStore outperforms the state-of-the-art models.

Index Terms—Data Analytics, Store Site Recommendation, Spatial Embedding, Deep Learning, Attention Mechanism

I. INTRODUCTION

In recent years, with the rapid development of internet technology and mobile devices, the amount of user-generated data from various sources in cities has grown explosively, such as social media, electronic commerce websites, mobile devices, sensor networks, etc. Urban computing exploits those big data generated by a diversity of sources in smart cities [1],[2] to tackle the major issues that cities face and provide the high-quality and beneficial services for the citizens [3], such as the fields of traffic [4], business [5] and so on. Store site recommendation is one of essential business services in smart cities for brick-and-mortar enterprises (e.g., the retail store). Choosing a good location when opening a new store is crucial for the future success of a business, since 94% of retail sales are still transacted in physical stores [6]. Therefore, an effective site recommendation system becomes necessary to brick-and-mortar enterprise managers.

The recent proliferation of multi-source data in cities has fostered unprecedented opportunities to the data-driven store site recommendation, which aims at leveraging large-scale user-generated data (e.g., check-in data, rating data) to analyze and mine users’ preferences for identifying the optimal location for a new store. Compared to traditional methods which generally conduct surveys to assess the value of store locations which is time-consuming and do not scale up well, the data-driven method can mine user behavior and extract knowledge from geographic datasets via data mining and machine learning techniques. For example, a lot of works [7-11] study the optimal location problem from location-based social networks, and most of them learn the regression model based on extracted features to predict the check-in numbers at given locations.

However, there still some limitations in present data-driven methods for store site recommendation. First, most works just analyze the popularity of the store location based on a single data source (e.g., check-in data), which lacks other significant data (e.g., real-word consumption data) affecting the store placement. Second, they rely on expertise feature engineering to characterize sophisticated influences, but it comes with a high cost to obtain high-quality features and hardly generalizes to other situations. Third, most models fail to learn complex feature interactions from multi-source data in complicated problems. Therefore, in this work, we aim to tackle abovementioned issues, and we study the problem of store site recommendation in a fine-grained manner.

Generally, the objective of the store site recommendation problem is to select the optimal location which can maximize sales of the store. Intuitively, different users play an important role in store site recommendation, since the overall sales of the store are decided by the consumption of each user. Therefore, to understand consumption behaviors of potential consumers at the store and help the selection of store location from the fine-grained perspective, we aim to predict the consumption level of different users at the store based on multi-source data (e.g., store data and user profiling data). Furthermore, we can infer whether the candidate location is appropriate to open the store in the long term in view of nearby customers.
The key challenge to solve this problem is modeling customer behaviors from multi-source data, including store data, user data and POI data. On the one hand, we need to extract valuable features from multiple types of data, such as user profile data and geographical data. On the other hand, we need to learn feature interactions among multiple features, since customer behavior is usually affected by various factors simultaneously.

Recently, deep neural networks (DNNs) [12] have experienced great success in many fields because of their great power of feature representation learning, such as computer vision [13] and natural language processing [14]. More importantly, DNNs have the ability to learn sophisticated feature interactions from raw data. More and more works have been trying to leverage DNNs to learn latent representations from raw data. For example, Lian et. al [15] propose a deep fusion model (DFM) including an inception module and an attention mechanism for feature-aware representation learning. Zhang et al. [16] study feature representations and propose Factorization-machine supported Neural Network (FNN).

However, these works mainly learn the feature representation from categorical data, which are not suitable to learn features from geographical data in our problem. In order to learn latent feature representation from multiple types of data in our work, we propose two embedding methods: the field embedding to reduce the dimensionality of sparse features (e.g., categorical data), the attention-based spatial embedding to learn the importance of spatial features.

A lot of works have been studying feature interactions without manual engineering. Factorization Machines (FM) is the typical framework which models the feature interactions as the inner product of latent vectors between different features. Although FM can achieve higher-order feature interactions theoretically, most FM methods only order 2 feature interactions due to high complexity. In view of the learning ability of neural networks, DNNs are used to model the high-order feature interactions implicitly [17]. To model both low- and high-order feature interactions, Cheng et al. [18] propose a typical hybrid network structure (Wide & Deep) that contains a linear (“wide”) model and a deep model, and it combines the benefits of memorization and generalization for recommender systems. However, the input of “wide part” still relies on expertise feature engineering. Further, some models are proposed to reduce the manual feature engineering, such as DeepFM [19], DCN [20] and xDeepFM [21]. For example, DCN contains a cross network which can efficiently capture feature interactions of bounded degrees. xDeepFM not only can learn explicit and implicit high-order feature interactions, but also can generate feature interactions at the vector-wise level. However, the major downside of these models is that they fail to learn latent feature representation and feature interactions from multi-source data. For instance, they do not consider the spatial distribution of POI data, and the effective interaction between dense features and sparse features.

In this paper, we design a unified interaction-aware model, named DeepStore, which can process different types of data from multiple data sources for store site recommendation. Our model is based on the Wide & Deep framework, which aims to efficiently capture low- and high-order feature interactions simultaneously. In particular, we design a novel cross network to learn high-order feature interactions explicitly from dense and sparse features. In addition, following the spirit of the DCN and xDeepFM models, we combine the explicit high-order interaction module (the cross network) with implicit interaction module (the deep network) and traditional linear module.

In summary, we make the following contributions:

- We formulate a problem for store site recommendation, which aims to predict the consumption level of different users at the store based on multi-source data.
- We propose a unified interaction-aware model based on the Wide & Deep network, named DeepStore, including the cross network module, the deep network module, and linear module. Unlike general Wide & Deep models [18],[19], it can jointly learn low- and high-order feature interactions explicitly and implicitly from multi-modal features.
- We design a new cross network to fuse multi-modal feature interactions explicitly. Specifically, it can model hybrid feature interactions, which makes sparse features interact at the vector level and dense features interact at the bite level.
- We propose two embedding methods in DeepStore, the filed embedding and attention-based spatial embedding, to learn latent feature representation from multi-modal data. In particular, two-level attention is designed in the attention-based spatial embedding to learn the importance of different spatial features in view of user profile and distance simultaneously.
- We evaluate our proposed model based on a real-world dataset, including store data, user data, and POI data. Extensive experiments are conducted from different perspectives, the results demonstrate that our DeepStore outperforms several state-of-the-art models.

The remainder of this paper is organized as follows. In Section II, we review the relevant work. Section III presents an overview of our proposed framework to solve the store site problem. In Section IV, we elaborate the proposed model, named DeepStore. The experiments based on real-world dataset are conducted in Section V. Finally, we conclude the article in Section VI.

II. RELATED WORK

In this section, we review the related work, including location recommendation, deep learning for feature interactions, and attention mechanism.

A. Location Recommendation

In recent years, location recommendation has been a trending research area, since the rapid increase of the availability of big data provides researchers with the possibility to access users’ location, such as check-in data. Point-of-interest (POI) recommendation and optimal site recommendation are two main research problems in location recommendation.

POI recommendation [22],[23] has become an important way
to help people discover attractive and interesting places, such as restaurants, hotels and so on. Yin et al. [24] propose a probabilistic generative model TRM for joint modeling of users’ check-in behaviors by exploiting the semantic, temporal and spatial patterns in a unified way. Liu et al. [25] propose an RNN-based neural network solution by modeling the user’s historical POI visits in a sequential manner. Feng et al. [26] propose a POI latent representation model, named POI2Vec, which incorporates the geographical influence of POIs to predict the potential visitors for a location in the next few hours. Qian et al. [27] propose a spatiotemporal context-aware and translation-based recommender framework (STA) to model the third-order relationship among users, POIs, and spatiotemporal contexts for large-scale POI recommendation. Yin et al. [28] propose a Spatial-Aware Hierarchical Collaborative Deep Learning model (SH-CDL), which models jointly performs deep representation learning for POIs from heterogeneous features and hierarchically additive representation learning for spatial-aware personal preferences.

Different from POI recommendation which recommends a place to people, site recommendation provides the optimal location for the enterprise to open the store. Early studies are based on dedicated models for store site recommendation. Sevtsuk et al. [29] analyze location patterns of retail and food establishments. It tests five hypotheses about retail locations found in previous literature using an economic model, and estimates the impacts of different location characteristics for store placement. Li et al. [30] present a modified Huff model to estimate the potential sales of individual Kmart and Wal-Mart stores. Tierno et al. [31] present a methodology for retail site location decision, which takes both geographic information systems and the analytical hierarchy process into consideration. However, those dedicated models mainly rely on domain expert knowledge or traditional data, which hardly learn related knowledge effectively under complex factors.

Recently, with the emergence of large-scale urban data, there is a potential to leverage these data to analyze and mine users’ preferences for store site recommendation. For example, Karamshuk et al. [8] demonstrate the power of geographic and user mobility features in predicting the best placement of retail stores based on check-in data. In [32], three types of features are incorporated into a regression model to predict the number of check-ins at a candidate location, including review-based market attractiveness features, review-based market competitiveness features, and geographic features. Lin et al. [33] analyze the popularity of a business location using Facebook data, and propose a model based on gradient boosting machine to estimate the popularity of a given target location. Xu et al. [34] propose a Demand Distribution Driven Store Placement (D3SP) framework for store location selection via mining search query logs of Baidu Maps. Guo et al. [11] propose a two-fold knowledge transfer framework based on collaborative filtering to solve the cold-start problem, which can transfer chain store knowledge from semantically-relevant domains.

Our work differs from previous works in the following aspects: First, the problem definition is different, we study the problem of store site recommendation in a fine-grained manner. Specifically, we aim to predict the consumption level of different users in the store, which not only can help the store placement, but also benefits to study customer behavior in the store at different time periods; Second, the dataset is different, we obtain real-world and fine-grained data to study customer behavior, including store data, user data and POI data. Third, the model for store site recommendation is different, we propose a unified interaction-aware model based on the neural network which can learn complex relations from multi-source data.

In this paper, we study customer behaviors in retail enterprises for store site recommendation. Different from traditional consumption behaviors in bricks-and-mortar stores, the retail enterprise in our research is a new business form, which combines online and offline business. That is to say, users could consume online and offline in the store simultaneously. The key challenge to solve this problem is modeling customer behaviors from multi-source data, including store data, user data and POI data. On the one hand, we need to extract valuable features and model feature interactions from multi-modal data. On the other hand, we need to model the influence of POIs with the spatial distribution. Therefore, in the following, we review some techniques to solve above-mentioned two issues, including deep learning for feature interaction and attention mechanism.

B. Deep Learning for Feature Interactions

Deep learning techniques have achieved great success in computer vision, speech recognition and natural language understanding. As a result, an increasing number of researchers are interested in employing DNNs to extract representative features from raw data and model deep interaction of features [35], [36].

Feature interaction based on deep learning has been widely studied in recommender system, especially in click-through rate (CTR) prediction. CTR prediction plays an important role in recommender systems, which aims at predicting the probability a user will click on a recommended item in view of complex factors. It is important for CTR prediction to model complex feature interactions behind user click behaviors, thus learning to extract features without manual engineering is necessary. To model both low- and high-order feature interactions, Cheng et al. [18] propose a typical hybrid network structure (Wide & Deep) that combines a linear (“wide”) model and a deep model. In this model, two different inputs are required for the “wide part” and “deep part”, respectively. However, the input of “wide part” still relies on expertise feature engineering. In order to reduce manual feature engineering, Guo et al. [19] propose a new neural network model DeepFM, which integrates the architectures of FM and deep neural networks. It can model low-order feature interactions like FM and models high-order feature interactions like DNN. However, Wide & Deep and DeepFM just model high-order feature interactions implicitly, since the function learned by DNNs can be arbitrary. Furthermore, Wang et al. [20] propose the Deep & Cross Network (DCN) model, which can learn high-order feature interactions implicitly and explicitly. Particularly, DCN
contains a cross network that can capture feature interactions of bounded degrees. However, DCN models feature interactions at the bit-wise level, which is different from the traditional FM framework which models feature interactions at the vector-wise level. The details about bit-wise and vector-wise feature interaction will be introduced in Section 4. Recently, Lian et al. [21] propose a new model, named xDeepFM, which can learn explicit and implicit high-order feature interactions effectively. In particular, they design a Compressed Interaction Network (CIN), which generates feature interactions in an explicit fashion and at the vector-wise level.

The abovementioned deep models, however, are mainly designed to learn representation and feature interactions from the data of a single modality. With the proliferation of multi-source data in cities, many efforts have been made to fuse multi-modal data and learn interactions among multi-modal features in the deep network. Nie et al. [37] present the multi-source mono-task learning model and its application in volunteerism tendency prediction based on aggregation of multiple social networks. In particular, they introduce multi-source dataset construction and how to effectively and efficiently complete the item-wise and block-wise missing data. Yin et al. [28] present a late feature fusion strategy into the SH-CDL model to deal with the multi-modal heterogeneous features of the POIs. Wu et al. Du et al. [38] propose a priority-based fusion method, which use exponential weights to model the overwhelming influences from the stronger content features. Wu et al. [39] propose a new module of Multi-modal Circulant Fusion to fully exploit interactions among multi-modal features. In particular, they define two types of interaction operations between original feature vectors and the reshaped circulant matrices.

C. Attention Mechanism

The attention mechanism has been successfully adopted in various machine learning tasks, which could improve the performance of the model. The main reason is that it can select valuable parts of the whole feature space. Xiao et al. [40] propose Attentional Factorization Machine, which learns the importance of each feature interaction from data via a neural attention network. Chen et al. [41] introduce the attention mechanism in CF to address the challenging item- and component-level implicit feedback in the multimedia recommendation. They propose an attention model including two attention components: the component-level attention module and the item-level attention module.

Visual attention is an effective method in computer vision [42],[43], which can learn representative spatial features from geographical data. Xu et al. [44] proposed the first visual attention model in image captioning, which automatically learns to describe the content of images. Xiao et al. [45] apply visual attention to fine-grained classification task using the deep neural network. Chen et al. [46] propose a convolutional neural network, named SCA-CNN, which combines spatial and channel-wise attentions in a CNN. Chen et al. [47] propose an attention-based configurable convolutional neural network (ABC-CNN) for visual question answering task, to locate the question-guided attention based on input queries.

In this paper, we will take full advantage of the deep models for feature interactions [20],[21]. In addition, inspired by the attention model in vision attention [46],[47], we also leverage the attention mechanism to learn better latent representations from a large number of geospatial data. In general, our model distinguishes the models in above-mentioned works in the following aspects. First, we design a unified interaction-aware model based on the Wide & Deep framework, which can process different types of data from multiple data sources. Second, we apply two embedding methods to learn dense and valuable features from categorical data and geographical data, respectively. Third, we propose two-level attention in the attention-based spatial embedding to learn the importance of different spatial features. Finally, we jointly learn low- and high-order feature interactions explicitly and implicitly from dense and sparse features simultaneously.

III. AN OVERVIEW

In this section, we will present the problem formulation, and describe the dataset we used, then analyze the complex factors that may influence the selection of store location. Finally, we illustrate the system framework of our work.

A. Problem Formulation

Store Site Recommendation Problem: Given some candidate locations to open a new store, the objective of the store site recommendation problem is to select the optimal location which can maximize sales of the store. On the one hand, from the coarse-grained perspective, we need to predict the overall sales of the store at the candidate location. On the other hand, from the fine-grained perspective, the consumption of different customers should be predicted to assist store site selection. Intuitively, different users play an important role in store site recommendation, since the overall sales of the store are decided by the consumption of each user. Therefore, to understand the consumption behaviors of potential consumers at the store and guide the selection of store location from the fine-grained perspective, we aim to predict the consumption situation of different users at the store. Further, we can infer whether the candidate location is appropriate to open the store in the long term in view of nearby customers.

User Consumption Behavior Prediction Problem: In general, the consumption behavior of a single user has a certain degree of randomness to some extent, which is affected by sophisticated factors in real situations. Therefore, we aim to predict the overall consumption of a group of people instead of predicting a single user’s consumption. Considering that geographical factor is one of the main impacts of consumer behavior in bricks-and-mortar stores, thus, we consider a group of people as the location-based community, which shares a sense of place that is situated in a given geographical area (e.g., a neighborhood). More specifically, a community in our dataset means a housing estate, which is a group of homes and other buildings built together.

Assume that there are m nearby communities around the candidate location Lq of the store, denoted as Cq = \{C_q^1, C_q^2, ..., C_q^l, ..., C_q^m\} . In each community, there are some
users that have the possibility to consume in the store, denoted as \( C_L = \{ u_1, u_2, \ldots, u_j, \ldots, u_n \} \). The user consumption behavior prediction problem aims to predict the users’ consumption of each community in the store at the candidate location \( L \). In view of users’ privacy, we just predict the level of users’ consumption rather than the amount of users’ consumption. Specifically, users’ consumption is divided into \( s \) levels based on equidistance partition, denoted as \( \{ y_1, y_2, \ldots, y_k \} \). Therefore, user consumption behavior prediction problem can be defined as the classification problem, which predicts the level of community users’ consumption in the store at the candidate location \( L \) during a given period of time (e.g., a month).

B. Dataset Description and Factors Analysis

In this work, we choose a retail enterprise for a case study, which is a chain retail enterprise owning multiple bricks-and-mortar stores in some cities across China. Different from traditional bricks-and-mortar stores, this retail enterprise is a new business form which combines online and offline business, known as the new retail [48]. Therefore, the consumption behaviors of users in these stores distinguish traditional consumption behavior. Fortunately, the combination of online business and offline business provides unique opportunities to analyze relationships between customer behaviors and multi-source data, such as online sales data and geospatial data. However, it also brings some new challenges, for example, how to analyze the complex factors that may influence the consumption of users in view of multi-source data, how to quantify the influence of multiple types of data on customer behavior, etc.

1) Dataset Description

The dataset in our work is the real-world commercial data, which is unique among existing works. Specifically, it includes three types of data for this work, including store data (e.g., location, sale information, customer visiting information), user profile data (e.g., age and gender), and POI data. Store data and user data sourced from the retail company, and POI data obtained from a Map platform. Table 1 gives a summary of the statistics of multi-source dataset we used in this work.

**Store data.** We mainly use two types of data in store dataset, including the profile information and historical sales data of the store. For the profile information, it contains the shop name, city, location (e.g., longitude and latitude), and opening time (e.g., year and month). For the historical sales data, each record contains customer id, shop name, and customer behavior (e.g., the level and time of consumption). Note that, in view of user privacy, we just obtain the overall consumption level of users in a community during a given period of time as the ground truth to evaluate the prediction result. For example, the consumption level of users in the community \( C_i \) in the store \( S_j \) is \( y_k^{ij} \) in February.

**User data.** Benefiting from the online business, we obtain user data that can establish the relationship with store sales data, which is unique comparing to existing store site recommendation works. Specifically, user data includes user location information and profile information. Similarly, to protect user privacy, we make statistics on user information in the community. For the location information, we obtain the number of people in a community and the location of the community. For the user profile, we have the following information: the number of men or women in the community, the number of people in different age groups, etc.

**POI data.** We rely on POI dataset to characterize the geographical environment of different places and further analyze the impact of the surrounding environment of bricks-and-mortar stores on customer behavior. In our work, POI data contains the information (e.g., name, location, and category) of multiple categories of POI related to the retail business, such as shop, food, transport facilities, and so on.

2) Multi-factor Analysis

There are various complex factors that may influence the consumption behavior of users in the store. Considering that the number of existing stores is small, in order to analyze customer behavior in detail, we treat each nearby community around the store as an instance instead of a store. Therefore, we measure the consumption level of users in each community in the store, which is used as the ground truth for the prediction model. In general, we take a community as an entity and analyze different factors which may influence customer behavior in the store based on three datasets. Specifically, these factors are classified into three categories, including user factors, geographic factors, and time factors.

**User factors.** Whether the user consumes in the store depends, to a great extent, on the user profile. For example, there is a higher possibility for women to consume in the clothing shop compared to men, families with babies would like to consume in the baby store. In addition, the amount of consumption of the user in the store is always associated with the user’s income level. Therefore, we make statistics on the number of people with different profiles in each community to characterize users in different communities. Specifically, user

<table>
<thead>
<tr>
<th>Data</th>
<th>Statistic information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store data</td>
<td>The number of stores</td>
</tr>
<tr>
<td></td>
<td>49</td>
</tr>
<tr>
<td>User data</td>
<td>The number of users</td>
</tr>
<tr>
<td></td>
<td>37,159,703</td>
</tr>
<tr>
<td>POI data</td>
<td>The number of POI categories</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>
factors contain the number of men or women in the community, the number of people in different age groups, the number of people with different professions, the number of people on different income levels, etc.

**Geographic factors.** The geographic factors [8],[28] we introduced assess spatial characteristics around the place where the store resides and the nearby communities. More specifically, we measure the following geographic factors of surrounding areas, which lie in a disk of radius \( r \) around the store and the community.

**Distance:** We consider the distance between the store located at \( l_i \) and the community located at \( l_j \), as formulated in (1). The farther away from the distance, the less likely customers go to the store.

\[
x_{ij}^{distance} = \text{Dist}(l_i, l_j)
\]

(1)

**Traffic convenience:** We use the number of transportation stations of multiple categories \( \Gamma_i \) (including bus stations and subway stations) in the surrounding area to denote the traffic convenience of the place at \( l_j \), which is defined as (2). Here, \( N^{ci}(l_j, r) \) is the number of transportation stations of category \( c_i \in \Gamma_i \) in the surrounding area, which is a disc centered at \( l_j \) with radius \( r \).

\[
x_{j}^{transport} = \sum_{c_i \in \Gamma_i} N^{ci}(l_j, r)
\]

(2)

**Density:** We calculate the total number of stores of different categories in surrounding areas to assess what extent the popularity of the place at \( l_j \), as defined in (3). Intuitively, a denser area could have a higher likelihood of attracting more users.

\[
x_{j}^{density} = N(l_j, r)
\]

(3)

**Neighbors Entropy:** We apply the entropy measure from information theory to measure the spatial heterogeneity and diversity of a place, which is defined as (4). We denote the number of place neighbors of type \( c_i \) with \( N^{ci}(l_j, r) \), where \( c_i \in \Gamma \) is one of the categories in POI data.

\[
x_{j}^{diversity} = -\sum_{c_i \in \Gamma} \frac{N^{ci}(l_j, r)}{N(l_j, r)} \log \frac{N^{ci}(l_j, r)}{N(l_j, r)}
\]

(4)

**Competitiveness:** We consider the competitive relationship between the stores belonging to the same category, which is measured as the proportion of neighboring places of the same type \( c \) with respect to the total number of nearby places, as formulated in (5).

\[
x_{j}^{competitiveness} = -\frac{N^{c}(l_j, r)}{N(l_j, r)}
\]

(5)

**Complementarity:** We consider the complementarity relationship of spatial interactions between different categories in the same area. We employ, Jensen Quality [7], defined by Jensen, to assess the complementarity relationship of spatial interactions of places with respect to their ability to attract other places of certain types. Specifically, it first uses a utility inter-type coefficient to quantify the dependency between different POI category pairs in the city. Then, the overall complementarity of the given place is computed based on the POI category distribution of the region, which is defined as (6) and (7).

\[
\rho_{yp-y_1} = \frac{N_{p}-N_{yp}}{N_{p} \times N_{yp}} \sum_{p} \frac{N_{y_1}(p,r)}{N_{y_1}(p,r)}
\]

(6)

\[
x_{j}^{com} = \sum_{y_{p} \in \Gamma} \log \left( \frac{N_{yp}(l,r)}{N_{yp}(l,r)} \right) (7)
\]

where \( y_p \) is the type of the place \( p \), \( N_{yp}(p,r) \) is the number of places of type \( y_1 \) in the surrounding area, which is a disc centered at \( p \) with radius \( r \), and \( N_{yp}(l,r) \) denotes how many places of type \( y_p \) are observed on average around the places of type \( y_1 \).

**POI set:** We also consider the number of POIs of related categories in the surrounding area of the place located in \( l_j \), which is defined as (8). In order to reflect the distribution of POIs, we divide the surrounding area into a set of location grids, and calculate the number of POIs in different location grids, where \( M^{ci}(l_j, r) \) is a matrix to represent the number of POIs of category \( c_i \) in different location grids.

\[
x_{j}^{POI} = \{M^{c1}(l_j, r), M^{c2}(l_j, r), \ldots, M^{cn}(l_j, r)\}
\]

(8)

**Time factors.** The time factors we explore attempt to capture customer behavior during different time periods. Brand awareness is one of the key factors that influence customer behavior in the store, and it constantly changes at different stages. Intuitively, we use the opening time of the store (e.g., the year) and the number of existing stores to evaluate the popularity of the brand in users. In addition, most of the user behaviors have the obvious seasonal characteristic, so we regard the month when users consume in the store as a factor to reflect the consumption habits. Moreover, the holiday is also an important factor which could stimulate consumption, such as New Year's Day and National Day. In addition to the time when users consume in the store, the opening time of the store.

### C. The DeepStore Framework

The framework of DeepStore is illustrated in Fig. 1, which mainly consists of five components: input features, embedding layer, cross network, deep network and combination layer.

**Input features.** Different from the computer vision or natural language understanding that the input data (e.g., images
or texts) can be applied directly to DNNs as the raw features, the multi-source data in our work is multi-dimensional and multi-categorical. Therefore, we extract some useful features from multi-source data based on the above factor analysis as the input features of DeepStore. Specifically, input features consist of four parts: expert features in purple box extracted from POI data, such as competitiveness and complementarity; store-related features in green box extracted from the store data, such as the opening time of the store; user-related features in blue box extracted from the user data, such as the number of people in different age groups; spatial features in red box extracted from the POI data, such as the distribution of POIs of related categories.

**Embedding layer.** The input features include sparse features and dense features. Considering that most sparse features are high-dimensional, so we employ an embedding layer to learn a low dimensional, dense real-value feature representation of sparse features. For the categorical feature which is always transformed into a high-dimensional sparse feature via one-hot encoding, we use the field embedding to reduce the dimensionality. For spatial features, we employ the attention-based spatial embedding to transform them into dense vectors.

**Cross network.** It exploits a feed-forward neural network to explicitly model the high-order feature interactions based on embedding features of sparse features and dense features, without any feature engineering besides raw features.

**Deep network.** It is a fully-connected neural network, which can learn implicit high-order interactions. It is noted that cross component and deep component share the same feature encoding.

**Combination layer.** To model both low-order and high-order feature interactions, we apply the combination layer to aggregate the linear competent, the cross network and the deep network to make the model stronger. Finally, based on the learned model, we can predict the level of user consumption in the store.

**IV. THE PROPOSED MODEL: DEEPSTORE**

In this section, we elaborate the proposed model for predicting the level of user consumption in the store. The DeepStore model starts with an embedding layer which is fed with the set of raw features extracted from multi-source data, followed by a cross network and a deep network in parallel. Finally, a combination layer is employed to combine the outputs from the linear component, the cross component, and the deep component. The complete DeepStore model is depicted in Fig. 1. A summary of the definition of the main notations used in our work is given in Table 2.

**A. Embedding Layer**

The input features of our model include sparse features and dense feature, which are extracted from multi-source data. Specifically, dense features refer to the vectors of real values, such as the distance, the number POIs, the number of people, etc. Sparse features in our work contain two types. The first one is the categorical feature (e.g., city and month), which are often encoded as one-hot vectors. The second one is the spatial feature, such as the POI set in different location grids. Generally, sparse features are high-dimensional compared to dense features. Therefore, we apply an embedding layer upon sparse features to learn a low-dimensional and dense feature vectors.

More specifically, we use two embedding methods in view of different characteristics of sparse features of two types. For the categorical feature, we use the field embedding to reduce the dimensionality. For spatial features, we employ the attention-based spatial embedding to transform them into dense vectors.

1) Filed Embedding

Recently, many works have used neural networks to learn advanced representation to replace sparse one-hot vectors. Following widely used methods in deep neural networks [20],[21], we adopt field embedding to learn a low-dimensional vectors.

Let $m$ denote the number of categorical feature fields, then the categorical features are encoded as high-dimensional sparse features via field-aware one-hot encoding, denoted as $x_r = [x_{r,1}, x_{r,2}, \ldots x_{r,i}, \ldots x_{r,m}]$, where $x_{r,i}$ is the binary vector in the $i$-th category. To reduce the dimensionality, we employ the field embedding to transform these binary features into dense vectors of real values. Let $x_{f,m} = [x_{f,m,1}, x_{f,m,2}, \ldots x_{f,m,1}, \ldots x_{f,m,m}]$ denote the output of the field embedding, and each field is mapped to a D dimension vector via a dictionary look-up operation:

$$x_{f,m,i} = W_{f,m,i} x_{r,i}$$

where $x_{f,m,i} \in \mathbb{R}^D$ denotes the embedding of the $i$-th field, and $W_{f,m,i}$ is the corresponding embedding matrix that will be optimized together with other parameters in the network. The field embedding is illustrated in Fig. 2.

2) Attention-based Spatial Embedding

Different from the categorical features, spatial features (e.g., the POI set in different location grids) have the spatial proximity relationship. In addition, spatial features of different grids and multiple categories have different influences on...
customers. Therefore, inspired by studies on the usage of attention mechanisms in computer vision [46,47], we propose the attention-based spatial embedding to learn the dense vectors of spatial features, as shown in Fig. 3.

**Input spatial features.** As mentioned in Section 3.2.2, in order to represent the distribution of geospatial locations of different POIs around the given place, we divide the surrounding area of the place into a set of location grids, and each of them has a size of \(k \times k\) (e.g., \(k = 500m\)). Therefore, the number of POIs in different location grids can be regarded as the image pixel data. For example, an image \(w \times h\) pixels means that there are \(w \times h\) grids in the surrounding area of the place. It is noted that the center of the area is the place, so \(w\) and \(h\) should be both even numbers. In addition, to characterize the influence of POIs of different categories, we consider \(C\) related categories in our POI data. Similarly, each category is considered as a channel of the image. In general, let \(X_c \in R^{W \times H \times C}\) denote input tensor of spatial features, where \(C\) is the number of POI category, \(w\) and \(h\) is the number of grids along the weight and height of the area, respectively.

Generally, spatial features have a different impact on customer behavior in different situations. For example, POIs closed to users may have an obvious influence on them compared to other remote POIs, and POIs of different category could attract different types of users. Therefore, we apply the attention mechanism in spatial embedding. Specifically, rather than representing the input raw feature into a static vector, the attention mechanism is able to learn the feature to evolve from the context in different situations, resulting in richer and longer representation for raw features. In general, we incorporate spatial-level attention and channel-level attention to learning the importance of spatial features in spatial embedding.

**Channel-level attention.** Different users may be attracted by POIs of different categories. Hence, we apply the channel-level attention to pay more attention to the useful POIs in view of different users. Given the input feature map \(X_s \in R^{W \times H \times C}\), we first use \(V = [v_1, v_2, ..., v_i, ..., v_c]\) to represent the original feature \(X_s\), where \(v_i \in R^{W \times H}\) is the \(i\)-th channel of the feature map. Note that we only focus on the influence on different channels in view of different user features \(f_t\), therefore, we ignore the spatial distribution of the feature map. Specifically, we apply mean pooling to each channel to obtain the channel feature \(v\), where \(v_i\) is the mean of \(v_i\), which represents the \(i\)-th channel features:

\[
v = [v_1, v_2, ..., v_i, ..., v_c] 
\]

(10)

The channel-level attention first uses a single-layer neural network to obtain the attention scores, then followed by a softmax function to generate the attention distributions \(\alpha\) over the grids. Formally, the channel-level attention network is defined as:

\[
A = \sigma(\langle W_{c1}\otimes v + b_{c1} \rangle + W_{sd}f_d) 
\]

(11)

\[
W_{c1} \in R^k, v \in R^C, b_{c1} \in R^k, W_{sd} \in R^{k \times p}, f_d \in R^p
\]

\[
\alpha = \text{softmax}(W_{c2}A + b_{c2}) 
\]

(12)

\[
W_{c2} \in R^k, A \in R^{k \times C}, b_{c2} \in R^1
\]

**Spatial-level attention.** In general, POIs closed to the given place (e.g., the store and community) may contribute more to customer behavior than other remote POIs. Therefore, feeding the global spatial features into the model may lead to suboptimal results due to other irrelevant regions. Rather than considering each grid of the area equally, the spatial attention could learn useful features from the important grids in view of the distance between the grid and the place. Let \(D \in R^{W \times H}\) denote the distance matrix between grids and the center, as the auxiliary features of the spatial-level attention.

Given the output \(X_{sc} \in R^{W \times H \times C}\) of the channel-level attention, we reshape \(X_{sc}\) to \(\mathbf{H} = [h_1, h_2, ..., h_i, ..., h_m]\) by flattening the width and height of the original feature \(X_{sc}\), where \(h_i \in R^C\) and \(m = W \times H\). Known the auxiliary feature \(D\), we also reshape it, denoted as \(f_d = [d_1, d_2, ..., d_i, ..., d_m]\), \(d_i \in R^1\). Similar to the definition of the channel-level attention, the spatial-level attention network is defined as:

\[
B = \sigma(\langle W_{s1}\otimes \mathbf{H} + b_{s1} \rangle + W_{sd}f_d) 
\]

(13)

\[
W_{s1} \in R^{k \times C}, \mathbf{H} \in R^{C \times m}, b_{s1} \in R^k, W_{sd} \in R^{k \times m}, f_d \in R^m
\]

\[
\gamma = \text{softmax}(W_{s2}B + b_{s2}) 
\]

(14)

\[
W_{s2} \in R^k, B \in R^{k \times m}, b_{s2} \in R^1
\]

**Output of attention-based spatial embedding.** Known the initial feature map \(X_s\), we can obtain the weighted feature map \(X_{scs}\) via the channel-level attention and spatial-level attention, calculated by (15). Finally, in order to retain the features of POIs of different categories, we use the fully-connected network for each channel to obtain the final embedding vector of spatial features instead of the convolutional network, which is defined as (16). Note that the spatial embedding of the \(i\)-th category of POI, \(x_{sm,i} \in R^D\) and field embedding of the \(j\)-th field, \(x_{fm,j} \in R^p\) are of the same length \(D\), and \(x_{sm} = [x_{sm,1}, x_{sm,2}, ..., x_{sm,l} ... x_{sm,c}]\).

\[
x_{scs} = f(X_s, \alpha, \gamma) 
\]

(15)

\[
x_{sm,i} = f(W_{sm,i}X_{scs,i} + b_{sm,i}) 
\]

(16)
Interactions, formulated as (18). However, traditional FM feature interactions at the vector level, such as the DNNs model. Specifically, feature interactions at the bit-wise level mean that the interaction occurs on an element rather than the whole feature vector. That is to say, even the elements within the same feature vector will influence each other.

Different from the traditional DNNs model which implicitly learns the high-order feature interactions explicitly at the bit-wise level, since an embedding vector is regarded as a unit for vector-wise interactions.

**B. Cross Network**

Cross network in our model explicitly learns the high-order feature interactions based on dense features and embedding of sparse features via a feed-forward neural network, without any other feature engineering. Before presenting the cross network in our model, we first introduce two interaction methods among features: bite-wise interactions and vector-wise interactions.

1) **Bite-wise Interactions**

In the deep learning-based recommendation, the multi-field categorical features are usually transformed into the high-dimensional and sparse features via field-aware one-hot encoding. To reduce the dimensionality, the embedding layer is widely used to transform these sparse features into dense vectors of real values, as mentioned above. Generally, the embedding features are directly fed into the feed-forward neural network to learn high-order feature interactions at the bit-wise level, such as the DNNs model. Specifically, feature interactions at the bit-wise level mean that the interaction occurs on an element rather than a whole feature vector. That is to say, even the elements within the same field embedding vector will influence each other.

Different from the traditional DNNs model which implicitly learns the high-order feature interactions, DCN can model the high-order feature interactions explicitly at the bit-wise level. In particular, DCN contains the Cross Network (CrossNet), whose hidden layers are calculated by the following cross operation:

\[
x_{i+1} = x_{i}x_{i}^{T}w_{i} + b_{i} + x_{i} = f_{i}(x_{i}, x_{i}, w_{i}, b_{i}) + x_{i}
\]  
(17)

where \( x_{i} \) and \( x_{i+1} \) are the outputs from the \( i \)-th and \((i+1)\)-th cross layers, respectively. \( w_{i} \) and \( b_{i} \) are the weight and bias parameters. The process is shown in Fig. 4. Assuming that the first two elements in \( x_{i} \) are the real values of dense features, others are elements of an embedding feature. We can see that the elements within the embedding feature will influence each other, however, it won’t happen to the dense feature since it only has one element.

2) **Vector-wise Interactions**

Traditional FM is the typical framework which models feature interactions at the vector-wise level, because the embedding vector is regarded as a unit for vector-wise interactions, formulated as (18). However, traditional FM hardly models the high-order feature interactions due to high computational complexity.

\[
y_{FM} = \langle w, x \rangle + \sum_{j=1}^{d} \sum_{j=2}^{d} (v_{j}, v_{j}) x_{j1}x_{j2}
\]  
(18)

\( w \in R^{d}, v_{j}, v_{j} \in R^{k} \)

On the basis of DCN and FM, xDeepFM is proposed which can jointly learn explicit and implicit high-order feature interactions effectively. Especially, xDeepFM includes a compressed interaction network (CIN) that learns high-order feature interactions at the vector-wise level, and the vector-wise interaction is illustrated in Fig. 5. Each layer in CIN has the following formula:

\[
x_{i+1} = \sum_{j=1}^{b_{i}} \sum_{j=1}^{m} W_{i}^{j} (x_{i-1} \circ x_{0}^{j})
\]  
(19)

where \( W_{i} \in R^{k \times m} \) is the parameter matrix for the \( h \)-th feature vector, and \( o \) denotes the Hadamard product. Different with CrossNet in DCN, the output of field embedding is transformed as a matrix \( X_{0} \in R^{m \times D} \), which is fed into the CIN to achieve feature interactions at the vector-wise level, since an embedding vector is regarded as a unit for vector-wise interactions.

3) **Hybrid Interactions**

For the deep learning-based recommendation, features explicitly interact at the vector-wise level is more appropriate than the bit-wise level, because most input features are sparse, of huge dimension. Therefore, we can define feature interactions at the vector-wise level based on the embedding features. However, there is no need to map a real value of the dense feature to a multiple dimension vector via the embedding method, which could increase the complexity of computation.

There are many both dense features and sparse features in our problem, in order to reduce the computational complexity, we propose the framework of hybrid interactions, which can make sparse features interact at the vector level, and dense features interact at the bite level, as shown in Fig. 6. Two types of features are fed into the cross network: the dense features from input features (indicated by the brown circles), denoted as a vector \( b_{o} \), and the embedding vectors from the embedding layer (indicated by the yellow circles), denoted as a matrix \( X_{0} \). Note the we first concatenate the output of field embedding \( x_{fm} \) and attention-based spatial embedding \( x_{sm} \), then formulate it as a matrix \( X_{0} \in R^{(m+c) \times D} \), where \( D \) is the dimension of the embedding, \( m, c \) is the number of categorical feature fields and POI category, respectively.

For each layer of the cross network, it first learns the feature interactions at vector level based on \( X_{0} \), which is defined in (20). The detailed computation procedure is illustrated in Fig. 5.

\[
x_{i+1} = f_{d}(X_{0}, X_{i}, W_{i})
\]  
(20)
Considering that the output of sparse feature interactions at the vector level is a matrix \( X_t \), and the output of dense feature interaction at the bite level is a vector \( b_t \). Therefore, we first apply sum pooling on \( X_t \), and then concatenate it and dense features as the input of feature interactions at the bite level. Finally, the output of the hidden layer \( b_t \) is calculated in (23).

\[
a_t = \text{sum pooling}(X_t)
\]

\[
g_t = \text{sum pooling}(f_d(X_{0:t-1}, W_{l-1}))
\]

(21)

\[
b_{t+1} = f_c(g_t, g_t, w_t^f, b_t^f) + x_t
\]

(23)

C. Deep Network

The deep network is a fully-connected feed-forward neural network, which is used to learn high-order feature interactions implicitly. The forward process is:

\[
h_{l-1} = \delta(W_{l-1}^e h_{l-1} + b_{l-1}^e)
\]

(24)

\[
h_l = \delta(W_l^h h_{l-1} + b_l^h)
\]

(25)

where \( x_t = [b_t, x_rtm, x_sen] \) is the output of the embedding layer, \( b_t \) is the dense features, \( x_rtm \) and \( x_sen \) are outputs of the field embedding and attention-based spatial embedding. \( h_l \) is the output of the \( l \)-th hidden layer in the \( l \)-th layer. \( \delta \) is the activation function.

D. Combination Layer

Inspired by the Wide&Deep and xDeepFM model, we apply the combination layer to combine the outputs from three components: linear component, cross network and deep network. On the one hand, the cross network and deep network can complement each other, since they learn the high-order feature interactions explicitly and implicitly respectively. On the other hand, linear component could learn linear relationships from raw features without the embedding method.

Therefore, the last hidden layer is the combination layer taking the following function:

\[
y_j' = \delta(w_{linear}^T l + w_{cross}^T h + w_{dnn}^T h + b_j)
\]

(26)

where \( l \) is the linear combination of raw features with different weights. \( h \) is the output of the deep network. In order to learn both low-order and high-order feature interactions, we concatenate the outputs of each hidden layer in the cross network, \( g = [g_0, g_1, ..., g_{L-1}] \).

The output of our proposed model is an s-way softmax which predict the probability distribution over s different levels of user consumption.

\[
\hat{y}_j = \frac{\exp(y_j')}{\sum_{j=1}^{s} \exp(y_j')}
\]

(27)

We define the loss function as the cross-entropy between labels \( y_{l,j} \) and the predicted results \( \hat{y}_{l,j} \) by (28), where \( N \) is the total number of training instances.

\[
L = -\frac{1}{N} \sum_{l=1}^{N} \sum_{j=1}^{s} y_{l,j} \log \hat{y}_{l,j}
\]

(28)

Finally, the optimization process is to minimize the following objective function in (29), where \( \Theta \) is the set of parameters in the model, \( \lambda \) is the regularization term:

\[
L(\Theta) = L + \lambda \| \Theta \|^2
\]

(29)

V. EXPERIMENTS

In this section, we first give a summary of the experimental purposes to evaluate the performance of our proposed model, DeepStore. Then, we introduce the experiment settings in our experiment. Next, we compare our proposed model and the other state-of-the-art models empirically. Finally, we make a discussion of the deep insights and limitations of our work.

A. Experimental Purposes

We conduct extensive experiments to answer the following questions:

- How does our proposed model perform as compared to the state-of-the-art methods?
- Can the attention mechanism effectively learn the importance of spatial features?
- How do the key hyper-parameters of our model impact its performance?
- Are there significant performance differences when choosing different locations in different cities?

B. Experimental Settings

1) Baseline Algorithms

We use 7 models as the baselines in our experiments, including Logistic Regression (LR), Gradient Boosting Decision Tree (GBDT), DNN, Wide&Deep, DeepFM, Deep&Cross Network (DCN), and xDeepFM. These models are highly related to our proposed model, and some of them are state-of-the-art models for deep learning-based recommender systems.

2) Evaluation Metrics

We measure the prediction performance of our model and baselines using the two metrics:

- **Accuracy**: Our problem is a multi-classification problem, so we use the accuracy to measure the performance of the prediction result. It calculates the number of correct predictions divided by the total number of predictions, as formulated in (30), where \( y_j \) is the real level of user consumption, \( \hat{y}_j \) is the predicted result. A higher accuracy means the better performance.

\[
\text{Accuracy} = \frac{\sum_{j=1}^{N} I(y_j = \hat{y}_j)}{N}
\]

(30)

- **Error**: Different from the general classification problem, the class in our problem represents the level of user consumption. Specifically, we divide the amount of consumption into \( s \) levels based on equidistance partition, denoted as \( \{y_1, y_2, ..., y_s\} \). Therefore, it also involves the relationship of size between different classes. For example, the difference between \( y_1 \) and \( y_4 \) is larger than is larger than that between \( y_1 \) and \( y_2 \). In general, error measures the average absolute difference between the \( y_j \) and \( \hat{y}_j \) by (31). A smaller error means the better performance.

\[
\text{Error} = \frac{\sum_{j=1}^{N} |y_j - \hat{y}_j|}{N}
\]

(31)

3) Parameter Setting and Training

Our models are learned by optimizing the loss function of (29), which is implemented using Tensorflow. Based on the empirical knowledge and our experiments, we first select
present the experimental results regarding C. hyper with λ = 0.001. Finally, to obtain the best prediction results, the learning rate is set to 0.001. Moreover, we use L2 regularization -batch stochastic optimization with performance of different models. Specifically, we apply mini-stores, and testing instances include the data of 10 stores.

To be fair, we use the same setting when comparing the performance of different models. Specifically, we apply mini-batch stochastic optimization with Adam optimizer, and the learning rate is set to 0.001. Moreover, we use L2 regularization with λ = 0.001. Finally, to obtain the best prediction results, the hyper-parameters of our model are tuned on the validation set, and the best settings will be shown in corresponding sections.

C. Experimental Results

Having depicted the experiment settings and baselines, we present the experimental results regarding the four experiment purposes given in Section 5.1.

1) Performance Comparison of Different Models

We want to compare the performance of different models. The result is shown in Table 3, where we have the following observations:

- We can find that deep models (e.g., DNN, Wide&Deep, DeepFM, DCN, xDeepFM and DeepStore) outperform traditional models (e.g., LR and GBDT), which demonstrates that the deep network has the advantage in capturing nonlinear relations from multi-source data.
- LR is far worse than all the rest models, since it only learns the linear relationship between labels and continuous features. However, therefore are some discrete features in our problem which leads to bad performance of LR.
- GBDT is a strong tree model which is widely used in many works, since it can deal with various types of data flexibly, including continuous and discrete features. In particular, it has strong robustness to noisy data. The data in our experiments is real data obtained from the store, which has a lot of noisy data. Therefore, GBDT can achieve better performance even compared with deep models.
- Wide&Deep, DCN, DeepFM, xDeepFM and DeepStore are significantly better than DNN, which directly reflects that, despite their simplicity, incorporating hybrid components are important for boosting the accuracy of predictive results.
- Another interesting observation is that Wide&Deep outperforms DeepFM and DCN. The differences among these three models are: Wide&Deep learns the feature interaction from initial extracted features, DeepFM and DCN model 2 and 4 feature interactions from embedding vectors in this experiment. Because the input data of deep models in our experiments contains some useful features which are extracted based on expert knowledge, so Wide&Deep could achieve better performance by exploiting useful features directly. Although DeepFM and DCN could learn high-order feature interaction, they may fail to acquire initial and useful features.
- xDeepFM and DeepStore outperform Wide&Deep, DeepFM and DCN, since they congregate advantages of Wide&Deep, DeepFM and DCN. More specifically, xDeepFM and DeepStore not only learn relationship from extracted features directly, but also model low-order and high-order feature interactions simultaneously.
- As we can see, our proposed model, DeepStore, achieves the best performance compared to all baselines, which demonstrates that combining explicit and implicit high-order feature interaction is necessary. Especially, DeepStore outperforms the xDeepFM, which is a state-of-the-art model in the deep learning-based recommendation. The results indicate that interactions over sparse and dense features are necessary, and the different embedding methods in view of different data types are important.
- Different from xDeepFM, our model has the ability to extract valuable features and model feature interactions from multi-modal data, such as learning representative spatial features from geographical data. However, the result of our method is close to the result of xDeepFM. The main reason that DeepStore gains less obvious improvement could be that the data used in our work suffers from scale and data sparsity issues, which impacts the significant improvement of the experimental performance. Specifically, we design various components (e.g., spatial embedding, cross
network, etc.) in the DeepStore model to extract valuable features from different types of data and learn complex feature interactions, because the store site recommendation problem is complicated, and customer behaviors are affected by sophisticated factors in real situations. This leads to more parameters to be learned in our model, which may also impacts the performance when the dataset scale is not big.

2) Impact of the Attention Mechanism

We want to know whether the attention mechanism effectively learns the importance of spatial features and improve the performance of the model. Fig. 7 shows the performance of DeepStore and its variants.

- **DeepStore** considers both channel attention and spatial attention in the attention-based spatial embedding.
- **DS_channel** just considers the channel attention, and **DS_spatial** considers the spatial attention.
- **DS_noattention** means that the fine-grained POI data is fed into the model without the attention mechanism.
- **DS_CNN1** means that there is one convolution layer instead of spatial embedding.
- **DS_CNN2** means that there are two convolution layers instead of spatial embedding.
- **DS_noPOI** means that there is not fine-grained POI data fed into the model.

First, we can see that DeepStore outperforms its variants, which demonstrates that the attention mechanism is effective in our problem to learn representative features from the fine-grained POI data. Furthermore, DS_channel and DS_spatial achieve better performance than DS_noattention, which indicates that both channel attention and spatial attention are useful in the spatial embedding. From Fig. 7, we can find that DS_CNN1 and DS_CNN2 perform worse than others in accuracy, because spatial features in our problem are different from features in the image. For example, CNN can extract features in different regions based on a filter. However, it is not suitable to extract spatial features in POI data, even POI data in our paper is regarded as the image pixel data. Because the given place stands in the center of the image, the distance between different regions and the given place plays the different role in extracting spatial features. Note that DS_noPOI performs worse than DeepStore, DS_channel and DS_spatial, but performs better than DS_noattention. This show that designing the efficient framework of the deep network is more important, compared to feeding all raw data into the deep network, which could lead to worse results.

![Fig. 7. Performance Comparison of DeepStore and its variants.](image)

**Fig. 7.** Performance Comparison of DeepStore and its variants.

3) Hyper-parameter Investigation

We study the impact of hyper-parameters on DeepStore, including the number of hidden layers, the number of neurons per layer, dropout and activation functions.

**Number of hidden layers.** Fig. 8 demonstrates the impact of the number of hidden layers in the deep network and the cross network. In Fig. 8(a), we can find that increasing number of hidden layers in the deep network improves the performance of models at the beginning. However, model performance degrades when the depth of the deep network is set greater than 3. This phenomenon is because of overfitting. Similarly, from Fig. 8(b), we can see that DeepStore achieves the better performance when the depth of the cross network is set to 4.

**Number of neurons per layer.** As shown in Fig. 9(a), model performance increases when the number of neurons per layer in the deep network is increased from 100 to 200, but it performs worse when we increase the number of neurons from 200 to 400. This is because an over-complicated model is easy to overfit. The increasing the number of neurons does not always bring benefit, as shown in Fig. 9(b), and 80 is a more suitable setting for the number of neurons per layer in the cross network.

**Dropout.** Dropout is a regularization technique to compromise the precision and complexity of the neural network. Fig. 10 shows the performance of the model when the dropout is set to different value. We can observe that our model achieves the best performance when the dropout is set to 0.8, because adding reasonable randomness to the model can strengthen the model’s robustness.
Improving results. Our work is a pilot study for fine-grained store site recommendation, where the problem definition is more difficult and the real-world commercial dataset is not easy to be obtained. First, it is quite difficult to predict the amount of user consumption, because customer behaviors have a certain degree of randomness which is affected by sophisticated factors in real situations. Therefore, the results in our experiments seem not as good as those of traditional coarse-grained problem definitions. Second, the real-world commercial data for fine-grained store site recommendation is quite difficult to be obtained and the commercial data used in our work is based on one type of new retail [48], which has grown in recent two years and the existing stores are limited in number and data quantity (see Table 1). In other words, the data used in our work suffers from data sparsity, scale, and noisy issues, which also affects the experimental performance. In order to improve the results, on the one hand, we intend to have more collaboration with commercial companies to gain data from maturing commercial entities, and on the other hand, we intend to optimize the prediction model to adapt to sparse data.

The Extension and Usage of the DeepStore to Other Applications. DeepStore is a deep model which can learn non-linear relations and high-order features interaction from multi-source data. Although our model is proposed to predict the level of user’s consumption, it is also applicable to other application, such as POI recommendation, CTR prediction. Different applications have different data characteristics, so we intend to extend our model and apply it to different application areas.

VI. CONCLUSIONS

In this paper, we propose a novel network named DeepStore, which aims to learn non-linear relations and high-order feature interactions from multi-source data. Particularly, DeepStore can automatically learn high-order feature interactions in both explicit and implicit fashions, which is of great significance to reducing manual feature engineering work. Finally, we conduct comprehensive experiments on the real-world dataset to compare the performance of DeepStore and state-of-the-art models from different perspectives. The results demonstrate the effectiveness of our approach. As for the future work, we intend to combine deep learning and transfer learning to transfer valuable knowledge from the city with rich data to the city with fewer data, and model the dynamic customer behavior based on the sequence model.

REFERENCES


