Dual Graph enhanced Embedding Neural Network for CTR Prediction

Wei Guo[1†], Rong Su[1†], Renhao Tan[1‡], Huifeng Guo[1], Yingxue Zhang[2], Zhirong Liu[1],
Ruiming Tang[1*], Xiuqiang He[1]

[1]Huawei Noah's Ark Lab [2]Huawei Noah's Ark Lab, Montreal Research Center

{guowei67,surong3,tanrenhao,huifeng.guo,yingxue.zhang,liuzhirong,tangruiming,hexiuqiang1}@huawei.com

ABSTRACT

CTR prediction, which aims to estimate the probability that a user will click an item, plays a crucial role in online advertising and recommender system. Feature interaction modeling based and user interest mining based methods are the two kinds of most popular techniques that have been extensively explored for many years and have made great progress for CTR prediction. However, (1) feature interaction based methods which rely heavily on the co-occurrence of different features, may suffer from the feature sparsity problem (i.e., many features appear few times); (2) user interest mining based methods which need rich user behaviors to obtain user's diverse interests, are easy to encounter the behavior sparsity problem (i.e., many users have very short behavior sequences). To solve these problems, we propose a novel module named **Dual Graph enhanced Embedding**, which is compatible with various CTR prediction models to alleviate these two problems. We further propose a **Dual Graph enhanced Embedding Neural Network** (**DG-ENN**) for CTR prediction. **Dual Graph enhanced Embedding** exploits the strengths of graph representation with two carefully designed learning strategies (divide-and-conquer, curriculum-learning-inspired organized learning) to refine the embedding. We conduct comprehensive experiments on three real-world industrial datasets. The experimental results show that our proposed **DG-ENN** significantly outperforms state-of-the-art CTR prediction models. Moreover, when applying to state-of-the-art CTR prediction models, **Dual graph enhanced embedding** always obtains better performance. Further case studies prove that our proposed **dual graph enhanced embedding** could alleviate the feature sparsity and behavior sparsity problems. Our framework will be open-source based on MindSpore[1] in the near future.

CCS CONCEPTS

• **Information systems → Recommender systems**;

---

[1]MindSpore. https://www.mindspore.cn/, 2020.

† Co-first authors with equal contributions, * Corresponding author, ‡Work done as intern at Huawei Noah's Ark Lab.

KEYWORDS

CTR Prediction, Embedding Enhancement, Graph Neural Network

1 INTRODUCTION

The prediction of click-through rate (CTR) plays a crucial role in many information retrieval (IR) tasks, ranging from web search, personalized recommendation and online advertising, which is multi-billion dollars business nowadays [14]. Most of the existing methods for CTR prediction can be classified into two categories, i.e., feature interaction modeling based methods and user interest mining based methods. Both feature interaction modeling based methods and user interest mining based methods follow a similar Embedding & Representation learning paradigm: input features are first transformed into trainable embedding vectors which are randomly initialized, and then transformed into fixed-length vector via feature interaction or interest mining, finally fed into fully connected layers to get the prediction score. Models in the former class such as Factorization Machines (FM) [23], Neural FM (NFM) [7], Product based Neural Network (PNN) [21] and FM based Neural Network (DeepFM) [6] focus on designing novel structure to capture more useful feature interactions more effectively. The models in the latter class like Deep Interest Network (DIN) [34], Deep Interest Evolution Network (DIEN) [33] and Multi-Interest Network with Dynamic routing (MIND) [14] aim to mine user interest from each user's behavior sequence precisely.

Though these two kinds of methods for CTR prediction have been investigated for years and obtained great progress, several challenges still exist, which limit the performance of existing methods, especially when deployed in large-scale industrial applications.

• **Feature Sparsity.** The performance of feature interaction based models heavily rely on the co-occurrence of different features. However, as the number of users and items growing continuously in real recommender system, there are numerous sparse features that appear very few times in the training set. To verify this discovery, we plot the feature frequency distribution of Tmall[2] and Alimama[3] dataset in Figure 1. As we can see, frequency of

---

[2]https://tianchi.aliyun.com/dataset/dataDetail?dataId=42
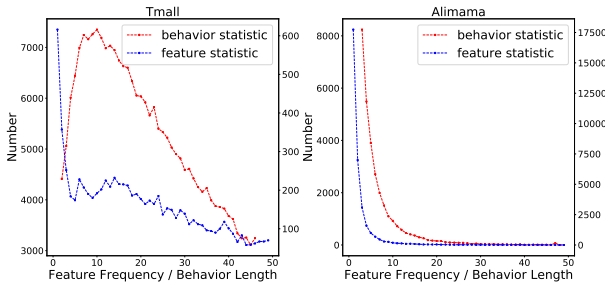[3]https://tianchi.aliyun.com/dataset/dataDetail?dataId=56

Figure 1: Statistics of feature frequency and behavior length distribution in Tmall and Alimama dataset.

most features are relatively low. It is hard for these methods to learn a good representation for these sparse features due to the low frequency of occurrence.

- **Behavior Sparsity.** User interest mining based methods need rich user behaviors to obtain user's diverse interests. However, user behaviors are characterized by heavy-tailed distributions, i.e., a significant proportion of users has very few interactions in the history, as shown in Figure 1. This poses a key challenge for these models with only limited user behavior information.

Recently, graphs have been used to represent relational information in recommendation datasets. Incorporating and exploiting the graph representation has shown to be effective for alleviating data sparsity [15, 28, 31]. The intuition motivating the involvement of graphs in recommender systems is that we include more relational information and increase the connectivity among users and items, leading to an improvement of recommendation quality.

We propose a novel Dual Graph enhanced Embedding Neural Network (DG-ENN), which is designed with two considerations to address the above two challenges in existing methods. Specifically, we construct two kinds of graphs (i.e., attribute graphs and collaborative graph) from different perspectives to tackle the two above-mentioned sparsity issues. *On the one hand*, a user (item) attribute graph is constructed by using user (item) features, such as gender, age, city, occupation (category, seller, brand). High-order proximity in the attribute graphs helps to enhance the embedding of sparse features, because learning embedding of a node also helps learning embedding of its neighbors, such that sparse features have more chances to be updated. With such enhanced feature embeddings, feature interactions can be learned more effectively. *On the other hand*, a collaborative graph is built from the collaborative signals between users and items. In this graph, there exist edges between a user and an item, representing the user interacting with this item. Besides such user-item edges, user-user edges are defined based on similarity of user profiles and behaviors while item-item edges are formulated based on their transition relations in user behaviors. Exploiting the proximity of this graph, user behaviors with short length can be enhanced with other users' behaviors, by learning node representations.

Yet, how to learn effective user and item representations from the two aforementioned graphs is still challenging due to the following two reasons. First, in the user (item) attribute graph, the user (item) attributes are of different fields with very different characteristics, which makes aggregating them during the learning process non-trivial. Second, in the collaborative graph, there are various kinds of

edges, resulting complex relations such as $u_1 \rightarrow v_1, u_1 \rightarrow u_2 \rightarrow v_1$, $u_1 \rightarrow u_2 \rightarrow v_2 \rightarrow v_1$ and $u_1 \rightarrow v_2 \rightarrow u_2 \rightarrow v_1$, which makes the relation modeling between user u_1 and item i_1 difficult. To handle such two issues, DG-ENN learns the user and item embeddings from these two graphs with two novel strategies. To learn embeddings in the user (item) attribute graph, a divide-and-conquer strategy is proposed to learn the information for each field of attributes individually and perform the information integration at the end, so that the information of different attributes (with different semantics) will not make the learning process chaotic. When learning from the collaborative graph, an organized learning mechanism, inspired by curriculum learning, is introduced to learn the user-user and item-item edges (which are relative easier to train) first, and learn user-item edges after that. Furthermore, DG-ENN serves as an embedding learning framework, which works compatibly with the existing deep CTR models, including both feature interaction modeling based and user interest mining based methods.

To sum up, our contributions in this paper can be summarized as follows:

- We propose a novel Dual Graph enhanced Embedding Neural Network named DG-ENN, which enhances the feature embedding in an end-to-end graph neural network framework. To the best of our knowledge, this is the first deep CTR model using graphs for alleviating the feature sparsity and behavior sparsity problems.
- More specifically, a user (item) attribute graph and a collaborative graph in DG-ENN are proposed to alleviate the feature sparsity and behavior sparsity problem. To learn these graphs effectively, we propose to perform a divide-and-conquer learning strategy and a curriculum-learning-inspired organized learning strategy for these two kinds of graphs, respectively.
- We perform extensive experiments on three public datasets, demonstrating significant improvements of DG-ENN over state-of-the-art methods for CTR prediction. The necessity of the two kinds of graphs is verified empirically. Moreover, the validity of the proposed two learning strategies is also demonstrated.

2 RELATED WORK

We briefly review three kinds of existing methods that are relevant to our work: 1) feature interaction modeling for CTR prediction, 2) user interest mining for CTR prediction, and 3) graph neural network for recommendation.

Feature Interaction Modeling for CTR prediction. Using raw features directly for CTR prediction can hardly lead to a good result, thus feature interactions modeling is playing a core role and has been extensively studied in the literature [17]. FM utilizes a low dimensional latent vector to represent each feature and learns 2-order feature interactions through the inner product of the related features' vectors [23]. Owing to its superior performance in learning feature interactions, many extensions of FM are proposed [10, 19, 30]. Recently, Deep Neural Network (DNN) has achieved great success with its great power of feature representation learning. It is promising to exploit DNN for CTR prediction. NFM [7] enhances FM with DNN to model non-linear and high-order feature interactions simultaneously. PNN further introduces a product layer between the embedding layer and DNN to model the feature

interactions [21]. Wide & Deep [3] and DeepFM [6] introduce an interesting hybrid architecture, which contain a shallow model and a DNN model to learn low-order and high-order feature interactions simultaneously. Deep & Cross Network (DCN) [26] and CIN [17] apply feature crossing at each layer explicitly. Thus the orders increase at each layer and are determined by layer depth.

User Interest Extraction for CTR prediction. Besides feature interactions modeling, user interest extraction is also very important. Many works are proposed recently that focus on learning user interest representation from user behavior history. DIN supposes that user interest is diverse, then uses an attention network to assign different scores to different user behaviors for user representation learning [34]. DIEN observes that user interest is dynamic, thus it utilizes GRU layers and auxiliary loss to capture evolving user interest for user's historical behavior sequence [33]. DSIN argues that user behavior sequence are composed of different homogeneous sessions [5]. So it employs self-attention layer and Bi-LSTM to model user's inter-session and intra-session interests. MIND learns multiple vectors for representing user's interests by using capsule network and dynamic routing architecture [14]. Despite great success has been made by these two kinds of CTR prediction methods, they cannot effectively solve the feature sparsity and behavior sparsity problems. We are going to solve them in this paper by incorporating and exploiting the graph representation learning.

Graph Neural Network for Recommendation. Graph Neural Network is widely used in recommender system in recent years. FiGNN models feature interactions via graph propagation on the fully-connected fields graph [16]. GIN utilizes user behaviors to construct a co-occurrence commodity graph to mine user intention [15]. GCMC [2] treats the recommendation task as a link prediction problem and employs a graph auto-encoder framework on the user-item bipartite graph to learn user and item embeddings. To better capture the collaborative signal existed in the user-item bipartite graph, many other GNN based works are then be proposed [8, 28, 31]. To make full use of other information beyond user-item interactions, KGAT [27] constructs a collaborative knowledge by combining user-item graph with knowledge graph and then applies graph convolution to get the final node representations. Heterogeneous graph Attention Network (HGAT) [18] utilizes a semantic-level attention network and a node-level attention network to discriminate the importance of neighbor nodes and node types. Although these GNN-based models have made progress, applying them directly for CTR prediction is still challenging, as depicted in Section 1.

3 PRELIMINARY

3.1 Problem Definition

In this section, we formulate the CTR prediction task with necessary notations. There are a set of M users $\mathcal{U} = \{u_1, u_2, ..., u_M\}$, a set of N items $\mathcal{V} = \{v_1, v_2, ..., v_N\}$, a set of J fields of user attributes $\mathcal{A} = \{\mathcal{A}^1, \mathcal{A}^2, ..., \mathcal{A}^J\}$, a set of K fields of item attributes $\mathcal{B} = \{\mathcal{B}^1, \mathcal{B}^2, ..., \mathcal{B}^K\}$ and a set of R fields of other features like timestamp, displayed position denoted as $C = \{C^1, C^2, \cdots, C^R\}$ to describe the context. The user-item interactions are denoted as a matrix $\mathcal{Y} \in \mathbb{R}^{M \times N}$, where $y_{uv} = 1$ denotes user u has interaction with item v before, otherwise $y_{uv} = 0$. Further, each user and item is associated with a list of attributes $\mathbf{A}_u \subset \mathcal{A}$ and $\mathbf{B}_v \subset \mathcal{B}$. In addition

to the attributes, each user also has a behavior sequence denoted as $\mathbf{S}_u = \{s_1, s_i, \cdots, s_{T_u}\}$, where $s_i \in \mathcal{V}$ and T_u is the length of user u's behavior sequence in the past. Besides user and item features, we denote context features as a list of $\mathbf{C} \subset C$. Concatenating all these features in a predefined order, one instance can be represented as:

$$\mathbf{x} = [u, v, \mathbf{A}_u, \mathbf{B}_v, \mathbf{S}_u, \mathbf{C}] \tag{1}$$

An encoding example of user ID, user attribute and user behavior feature is presented as:

$$\underbrace{[0, 0, \cdots, 1, 0]}_{u : UserID} \qquad \underbrace{[0, 1, \cdots, 1]}_{\mathbf{A}_u : Occupation\&City} \qquad \underbrace{[1, 1, \cdots, 1, 0]}_{\mathbf{S}_u : Click}$$

The representations of other features are similar, so we omit them for simplicity. It is noticed that we categorize user id, item id, user attributes, item attributes and context as features except the user behavior. These features may encounter the feature sparsity problem when appear very few times. The goal of CTR prediction is to predict the probability that user u will be interested in the target item v under context $\mathbf{C}$.

3.2 Base model

Most of existing CTR prediction methods follow a similar Embedding & Representation learning & Prediction paradigm. We refer them as the base model in this section.

3.2.1 Initial Embedding. The input data in CTR prediction are usually in a high-dimensional sparse binary form. It is common to apply an embedding layer upon the input to compress it into a low dimensional, dense real-value vector by looking up from an embedding table $\mathbf{W}$. For one-hot vector u, v, the embedding representation is a single vector. For multi-hot vector $\mathbf{A}_u, \mathbf{B}_v, \mathbf{S}_u, \mathbf{C}$, the embedding representation is a list of vectors. The embedding vectors of these fields are then concatenated together to get the embedding of the whole input features.

$$\mathbf{E} = [\mathbf{e}_u, \mathbf{e}_v, \mathbf{e}_{\mathbf{A}_u}, \mathbf{e}_{\mathbf{B}_v}, \mathbf{e}_{\mathbf{S}_u}, \mathbf{e}_{\mathbf{C}}] \tag{2}$$

3.2.2 Representation Learning. Many existing works focus on designing advanced network architecture for feature interaction modeling or user interest mining, which can be formulated as:

$$\mathbf{P} = f(\mathbf{E}) \tag{3}$$

For simplicity, we use the inner product module as the base representation learning module:

$$\mathbf{P} = [\mathbf{E}_1, \cdots, \mathbf{E}_F, \langle \mathbf{E}_1, \mathbf{E}_2 \rangle, \cdots, \langle \mathbf{E}_{F-1}, \mathbf{E}_F \rangle] \tag{4}$$

where $\langle, \rangle$ denotes the inner product operation, F is the number of fields. We also evaluate the performance of other representation learning module in the experiment section to validate the effectiveness of our proposed dual graph enhanced embedding.

3.2.3 Fully Connected Layer. The output of representation learning component is fed into the fully connected layer, which serves as a classifier.

$$\mathbf{a}^{(l)} = \sigma(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}) \tag{5}$$

where $\mathbf{a}^{(0)} = \mathbf{P}$, l is the current layer depth and σ is the activation function. $\mathbf{a}^{(l-1)}$, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the input, model parameters and

bias of the l-th layer. The output is a real number as the predicted CTR:

$$\widehat{y} = Sigmoid(\mathbf{W}^{(L)}\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)}) \qquad (6)$$

3.2.4 Model Training. The widely-used logloss is adopted as the objective function, which is defined as:

$$L_{logloss} = -\frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x},y)\in\mathcal{S}} y(\mathbf{x})log\widehat{y}(\mathbf{x}) + (1 - y(\mathbf{x}))log(1 - \widehat{y}(\mathbf{x})) \qquad (7)$$

where $|\mathcal{S}|$ is the total number of training instances, $y(\mathbf{x})$ is the real value for input vector $\mathbf{x}$, and $\widehat{y}(\mathbf{x})$ is the predicted value by our model.

4 DUAL GRAPH ENHANCED EMBEDDING

As stated earlier, most of existing methods focus on the representation learning layer, while overlook the embedding layer. Whereas, embedding layer with random initialization suffers from the feature sparsity and behavior sparsity issue. Motivated by this observation, in this paper, we focus on the embedding learning with a dual graph enhanced embedding network (DG-ENN) based on the base model.

Dual graph enhance embedding component contains three modules: **graph construction**, **attribute graph convolution** and **collaborative graph convolution**. In this section, we elaborate each of these three modules in detail. Figure 2 gives a depiction of our proposed dual graph convolution framework.

4.1 Graph Construction

4.1.1 Attribute Graph. An attribute can be in multiple users or items, serving as a bridge to improve their representation. Based on this bridge, we construct two attribute graphs $\mathcal{G}_{ua} = \langle \mathcal{U} \cup \mathcal{A}, \mathcal{E}_{ua} \rangle$ and $\mathcal{G}_{vb} = \langle \mathcal{V} \cup \mathcal{B}, \mathcal{E}_{vb} \rangle$. Edges $e_{ua} = (u, a)$ and $e_{vb} = (v, b)$ indicate that attribute a belongs to user u and attribute b belongs to item v. The attribution graphs establish attribute connections to alleviate the feature sparsity problem.

4.1.2 Collaborative Graph. Inspired by the collaborative filtering (CF) that similar users may exhibit similar preference on items [24], we utilize the collaborative signals to expand user behaviors and therefore alleviate the behavior sparsity problem. User-item interactions matrix $\mathcal{Y}$ can be regarded as a user-item bipartite graph $\mathcal{G}_{uv} = \langle \mathcal{U} \cup \mathcal{V}, \mathcal{E}_{uv} \rangle$. There is an edge $e_{uv} = (u, v)$ if $y_{uv} = 1$. However, $\mathcal{G}_{uv}$ only reveals the user-item interaction relation, but ignores the direct connections inside users and inside items. As a result, we construct user-user similarity graph and item-item transition graph to extract such more complex relations. The user-user similarity graph is built based on the user preferences and user attributes simultaneously:

$$sim(i, j) = \alpha_1 \frac{\langle \mathbf{Y}_i, \mathbf{Y}_j \rangle}{||\mathbf{Y}_i|| \cdot ||\mathbf{Y}_j||} + \alpha_2 \frac{\langle \mathbf{A}_i, \mathbf{A}_j \rangle}{||\mathbf{A}_i|| \cdot ||\mathbf{A}_j||} \qquad (8)$$

where $\mathbf{Y}_i$ and $\mathbf{Y}_j$ denote the i-th and j-th row of the user-item interaction matrix $\mathcal{Y}$, $\mathbf{A}_i$ and $\mathbf{A}_j$ denote the attributes of corresponding user i and user j. We set $\alpha_1 = \alpha_2 = 0.5$ for simplicity. After calculating the overall similarity between each two users, we can build a k-NN graph $\mathcal{G}_{uu} = \langle \mathcal{U} \cup \mathcal{U}, \mathcal{E}_{uu} \rangle$ with a pre-defined k. The item-item transition graph is built based on the sequential information of different users' behavior sequences. Two items are connected in the item-item transition graph if they are interacted by the same users consecutively. With all users' behavior sequences considered, we can construct an item-item graph $\mathcal{G}_{vv} = \langle \mathcal{V} \cup \mathcal{V}, \mathcal{E}_{vv} \rangle$. It can reflect user's preferences on group of items, which are ignored by the user-item bipartite graph. As a result, we can get the overall user-item collaborative graph $\mathcal{G}_{cf} = \langle \mathcal{U} \cup \mathcal{V}, \mathcal{E}_{uu} \cup \mathcal{E}_{uv} \cup \mathcal{E}_{vv} \rangle$. By aggregating neighborhood information from $\mathcal{G}_{cf}$ iteratively, user's representation can be enhanced with other users' behaviors, thus the behavior sparsity problem can be alleviated.

4.2 Attribute Graph Convolution

With the two attribute graphs $\mathcal{G}_{ua}$ and $\mathcal{G}_{vb}$, we enrich the representation of sparse features with graph convolution. The user (item) attributes contain different fields with very different characteristics (for example, item price and item category are very different in semantics as well as distributions), which makes aggregating them during the learning process non-trivial.

However, most of existing GNNs mix neighbors information indistinguishably and fail to distinguish different characteristics of neighbor attributes nodes, leading to sub-optimal results [8, 12, 28]. To consider different characteristics of attributes, we propose a divide-and-conquer strategy to integrate different attribute information while maintaining their intrinsic characteristics. More specifically, we learn the information for each field of attributes individually and perform information integration at the end.

4.2.1 Field-wise Information Propagation. We first describe the information propagation within a field of attributes. We adopt the state-of-the-art GCN models for such field-wise information propagation. We use h to denote the central node and N_h to denote its neighbor set in this graph. We adopt the following three types of GCN aggregators as potential candidates:

- *GCN Aggregator.* GCN [12] sums up the representation of central node and its neighbors and then applies a nonlinear transformation to generate the new representation:

$$f_{GCN}^{(l)}(\mathbf{e}_h^{(l)}, \mathbf{e}_{N_h}^{(l)}) = \sigma(W^{(l)} \sum_{i\in\{h\}\cup\{N_h\}} d(h, i)\mathbf{e}_i^{(l)}) \qquad (9)$$

where $\mathbf{e}^{(0)}$ is the initial embedding from $\mathbf{E}$, σ and $W^{(l)}$ are the nonlinear activation function and transformation matrix of layer l. $d(h, i) = 1/\sqrt{|N_h||N_i|}$ is the normalization factor.

- *NGCF Aggregator.* NGCF [28] improves GCN by considering additional feature interactions between central node and neighbor nodes. Besides, it aggregates the neighbors first and then add the neighbor representation to central representation, which can be formulated as follows:

$$f_{NGCF}^{(l)}(\mathbf{e}_h^{(l)}, \mathbf{e}_{N_h}^{(l)}) = \sigma(W_1^{(l)}\mathbf{e}_h^{(l)} + \sum_{i\in N_h} d(h, i)(W_1^{(l)}\mathbf{e}_i^{(l)} + W_2^l(\mathbf{e}_h^{(l)} \odot \mathbf{e}_i^{(l)}))) \qquad (10)$$

where $W_1^{(l)}$, $W_2^{(l)}$ are the trainable weight matrix and $\odot$ denotes the element-wise product.

- *LightGCN Aggregator.* LightGCN [8] argues that the feature transformations and nonlinear activation function are not necessary
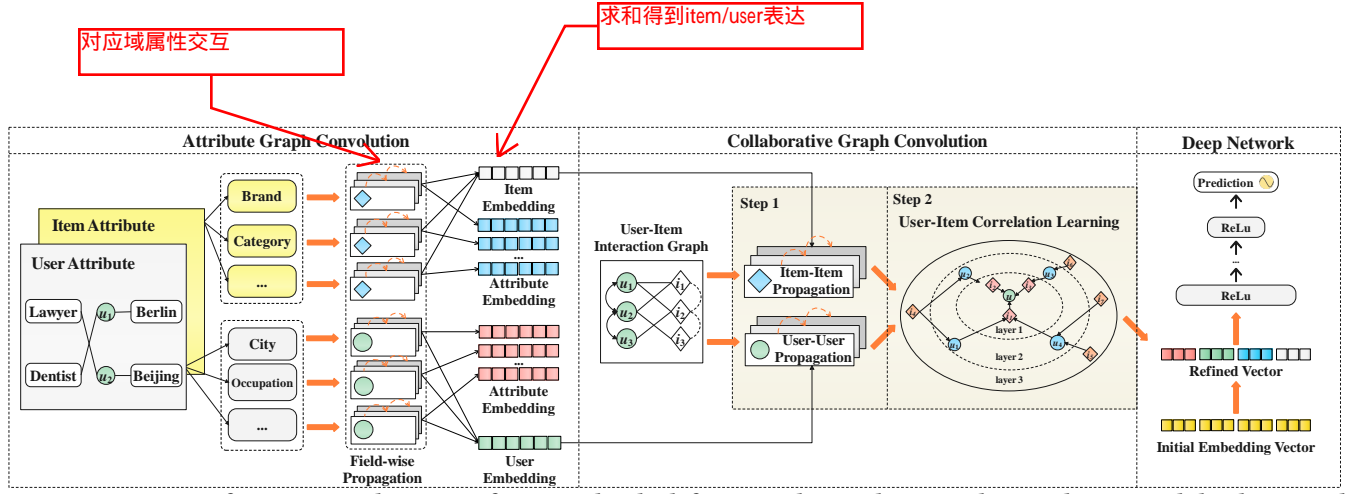
Figure 2: Overview of our proposed DG-ENN framework. The left part is the attribute graph convolution module, the central part is the collaborative graph convolution module and the right part is the deep network module.

and might even degrade the recommendation performance. Therefore, it removes the weight matrix and activation function:

$$f_{LightGCN}^{(l)}(\mathbf{e}_h^{(l)}, \mathbf{e}_{N_h}^{(l)}) = \sum_{i \in N_h} d(h,i)\mathbf{e}_i^{(l)} \qquad (11)$$

The feature representation with layer $l+1$ information propagation is formulated as:

$$\mathbf{e}_h^{(l+1)} = f_\star^{(l)}(\mathbf{e}_h^{(l)}, \mathbf{e}_{N_h}^{(l)}) \qquad (12)$$

Noticed that we use separate parameters for different fields when using GCN or NGCF aggregators. As aggregators are very important for the performance of our method, we will evaluate the effectiveness of the three GCN aggregators in experiment section. After propagation of L layers, we have $L+1$ embeddings for each node. Following [8], we average these embeddings to get the final embedding for all central nodes:

$$\hat{\mathbf{e}}_h = \sum_{l=0}^{L} \mathbf{e}_h^{(l)} \qquad (13)$$

4.2.2 Cross-field Information Integration. As we have J fields of user attributes and K fields of item attributes, we generate J user representations and K item representations by field-wise information propagation in the previous section. As different fields of attributes have different importance to the final representation, it's natural to employ an attention mechanism to assign different importance scores for individual representations. However, as the main contribution of this part of model is introducing the attribute graphs and modeling field-wise information individually (the effectiveness of which will be validated empirically in the experiments), we apply the average operation over multiple embeddings to get the final user and item representations:

$$\mathbf{z}_u = \sum_{t=1}^{J} \hat{\mathbf{e}}_u^{(t)}, \mathbf{z}_v = \sum_{t=1}^{K} \hat{\mathbf{e}}_v^{(t)} \qquad (14)$$

where $\hat{\mathbf{e}}_u^{(t)}$ denotes the embedding obtained from Section 4.2.1 with respect to user attribute of field t (we omit t in Section 4.2.1 for the sake of clarity). The embedding of all the features in data instance can be refined as:

$$\mathbf{Z} = [\mathbf{z}_u, \mathbf{z}_v, \mathbf{z}_{\mathbf{A}_u}, \mathbf{z}_{\mathbf{B}_v}, \mathbf{z}_{\mathbf{S}_u}, \mathbf{e}_C] \qquad (15)$$

Noted that all the features (except contextual features) are enhanced. The reason why we don't construct graphs for contextual features is the risk of introducing noise as there are no clear relations between users/items and contextual features in most cases.

4.3 Collaborative Graph Convolution

The behavior sparsity issue is a challenge for the model to capture user interests with very limited user behavior information. Using the high-order proximity of the collaborative graph to enrich user behaviors is beneficial to alleviate this issue. However, the underlying reasons motivating a user to click an item may be various, which might be difficult for existing models to capture such complex relations. For example, $u_1 \to v_1$, $u_1 \to u_2 \to v_1$, $u_1 \to u_2 \to v_2 \to v_1$ and $u_1 \to v_2 \to u_2 \to v_1$ are all possible reasons to drive user u_1 to click on a target item v_1. Existing meta-path based methods, like [9, 29], introduce additional information with the path extraction strategy. However, they need expert knowledge to design meta-paths. Besides, it's difficult for meta-path methods to exhaustively search all useful meta-paths, which largely limits their performance. GNN based methods use neighbor aggregation for behavior expanding, which don't need domain knowledge. However, existing GNN based methods [18, 27, 32] aggregate different types of neighbors at the same time, which overlook the dependency during the process of neighbor aggregation. This reduce their ability to model graph correlations and more complex graph structure. To solve these issues, we design an organized learning mechanism by taking inspiration from the curriculum learning, which introduces different concepts at different time and then uses the previous learned concepts to promote the learning of new concepts [1]. Concretely, we first learn separate representations for users and items using user-user edges and item-item edges, then we use the user-item edges to learn the correlations between users and items. By this way, the complex node relation can be modeled well. As shown in the right part of Figure 2, collaborative graph convolution network includes two components: 1) information propagation within Users/Items, 2) behavior expanding across users and items.

4.3.1 Information Propagation within Users/Items We first illustrate the information propagation within users/items. The input of this component is the refined embedding from attribute graph convolution module. Taking user node as an example, we denote the central node as u and its user neighbor set as N_u. The information

propagation is formulated as:

$$\mathbf{z}_u^{(l)} = f_\star^{(l-1)}(\mathbf{z}_u^{(l-1)}, \mathbf{z}_{N_u}^{(l-1)}) \qquad (16)$$

where $\mathbf{z}_0^{(0)}$ and $\mathbf{z}_{N_u}^{(0)}$ are the refined embedding from Z. We use the average pooling of all layer's output as the final representation, as different layers of information propagation can represent different length of relations:

$$\hat{\mathbf{z}}_u = \sum_{l=0}^{L} \mathbf{z}_u^{(l)} \qquad (17)$$

Similarly, the representation for item nodes is:

$$\hat{\mathbf{z}}_v = \sum_{l=0}^{L} \mathbf{z}_v^{(l)} \qquad (18)$$

4.3.2 Behavior Expanding Across Users and Items. After learning from user-user and item-item edges, we use user-item edges to learn the user-item preferences that can be used for user behavior expanding. Taking user node as an example, the user-item correlations can be modeled as:

$$\mathbf{q}_u^{(l)} = f_\star^{(l-1)}(\mathbf{q}_u^{(l-1)}, \mathbf{q}_{N_u}^{(l-1)}) \qquad (19)$$

where $\mathbf{q}_u^{(0)} = \hat{\mathbf{z}}_u$ and $\mathbf{q}_{N_u}^{(0)} = \hat{\mathbf{z}}_{N_u}$ are the enriched embedding after information propagation within users/items. Then we use the average pooling of all layers' output as the final representation:

$$\hat{\mathbf{p}}_u = \sum_{l=0}^{L} \mathbf{p}_u^{(l)} \qquad (20)$$

Similarly, the embedding of item v is generated by the same process:

$$\hat{\mathbf{p}}_v = \sum_{l=0}^{L} \mathbf{p}_v^{(l)} \qquad (21)$$

Notice that $\mathbf{q}_u^{(0)} = \hat{\mathbf{z}}_u$ and $\mathbf{q}_v^{(0)} = \hat{\mathbf{z}}_v$ are also included in the final representation, because user-user relations and item-item relations also contain useful neighbors that can be used to expanding user behaviors. Comparing with equation 2, after the graph enhanced operations, we can get the final enhanced embdding for all the features:

$$\mathbf{P} = [\hat{\mathbf{p}}_u, \hat{\mathbf{p}}_v, \mathbf{z}_{A_u}, \mathbf{z}_{B_v}, \hat{\mathbf{p}}_{S_u}, \mathbf{e}_C] \qquad (22)$$

4.4 Complexity Analysis

Since scalability is important for graph-based algorithms, we analyze the time complexity of DG-ENN for model training and online inference respectively. As the enhanced embedding can be used directly for online inference, the time complexity of DG-ENN is **the same as** base model. For model training, the layer-wise graph convolution is the main time cost. Taking LightGCN aggregator as an example, the computational complexity for attribute graph is $O(L \cdot |\mathcal{G}_{ua}| \cdot d)$, where $|\mathcal{G}_{ua}|$ denotes the number of edges existed in $\mathcal{G}_{ua}$, d is the embedding size and L is the number of graph convolution layers. Similarly, the computational complexity for collaborative graph is $O(L \cdot |\mathcal{G}_{cf}| \cdot d)$. In real-world industrial application, there may be numerous edges connecting users (items) with attributes and connecting users and items. To scale up the model training, neighbor sampling is necessary.

Table 1: Dataset statistics.

Dataset	#Users	#Items	#Instances	#Features	#Fields
Alipay	438,380	800,496	4,822,180	1,248,930	5
Tmall	415,800	565,888	4,573,800	994,771	8
Alimama	43,047	47,240	473,517	158,338	12

5 EXPERIMENTS

5.1 Experiment Setup

5.1.1 Datasets. We evaluate the effectiveness of our proposed model on three large-scale datasets: *Alipay*, *Tmall*, and *Alimama*.

- **Alipay**[4]: This dataset is provided by Ant Financial Services in IJCAI-16 contest [20]. It contains users' online/on-site behavior logs in 2015. Each log contains multiple fields, including user ID, item ID, seller, category, online action type and timestamp.
- **Tmall**[5]: This dataset is provided by Tmall.com in IJCAI-15 contest [20]. The user profile is described by user ID, age range and gender. The item attributes include category and brand. The context features are timestamp and action type.
- **Alimama**[6]: This dataset is provided by Alimama [5]. Each log in this dataset is composed of 12 feature fields including user ID, item ID, user micro group ID, occupation, shopping level, brand, category and some other information.

5.1.2 Dataset Preprocessing. For each user, their clicked items are sorted by the interaction timestamp. Following [20, 22], we split the dataset for evaluation. Specifically, supposing there are **T** historical behaviors for a user, behavior [*1, T-3*] are collected as user behavior feature in the training set to predict the target item *T-2*. Similarly, behavior [*1, T-2*] are used as user behavior feature in the validation set to predict the target item *T-1*, behavior [*1, T-1*] are used as user behavior feature in the testing set to predict the target item *T*. For each user, we random sample 10 non-clicked items to replace the target item as the negative samples. Table 1 shows the statistics of the three datasets.

5.1.3 Baseline Models. To verify the effectiveness of our proposed DG-ENN framework, we compare it with three groups of CTR prediction models: (A) feature interaction based models (LR [13], FM [23], DeepFM [6], PNN [21], AutoINT+ [25]); (B) user interest mining based models (DIN [34], DIEN [33]); (C) GNN based models (GIN [15], FiGNN [16]).

5.1.4 Evaluation Metrics. We adopt two widely-used evaluation metrics, namely *AUC* and *Logloss* [6], to evaluate the performance. *AUC* (↑) measures the goodness of assigning positive samples higher scores than randomly chosen negative samples. A higher AUC value indicates a better performance. *Logloss* (↓) measures the distance between the predicted scores and the true labels. A lower Logloss value means a better model performance.

5.1.5 Parameter Settings. For fair comparison, we set embedding dimension of all models as 10, and batch size as 2000. We tune learning rate from {1e-1,1e-2,1e-3,1e-4}, L_2 from {0,1e-1,1e-2,1e-3,1e-4,1e-5}, and dropout ratio from 0 to 0.9. The deep layers for all models are {400,400,400,1}. The models are optimized with Adam

Table 2: The overall comparison. ⋆ indicates a statistically significant level p-value<0.05 comparing DG-ENN with the best baseline (indicated by underlined numbers).

Dataset	Alipay		Tmall		Alimama	
Model	AUC	Logloss	AUC	Logloss	AUC	Logloss
LR	0.8196	0.2276	0.8760	0.1991	0.7207	0.2693
FM	0.8498	0.2175	0.9026	0.1831	0.7396	0.2668
AutoInt+	0.8631	0.2147	0.9181	0.1730	0.7499	0.2611
DeepFM	0.8648	0.2084	0.9155	0.1774	0.7653	0.2581
PNN	0.8756	0.2020	0.9261	0.1650	0.7758	0.2534
DIN	0.8649	0.2081	0.9169	0.1761	0.7644	0.2584
DIEN	0.8731	0.2037	0.9235	0.1684	0.7710	0.2554
GIN	0.8645	0.2093	0.9194	0.1716	0.7621	0.2595
FiGNN	0.8632	0.2121	0.9180	0.1753	0.7438	0.2635
DG-ENN	**0.9216⋆**	**0.1674⋆**	**0.9501⋆**	**0.1399⋆**	**0.8443⋆**	**0.2254⋆**

Table 3: Compatibility of embedding enhancement.

Dataset	Alipay		Tmall		Alimama	
Model	AUC	Logloss	AUC	Logloss	AUC	Logloss
PNN	0.8756	0.2020	0.9261	0.1650	0.7758	0.2534
DG-PNN	**0.9216**	**0.1674**	**0.9501**	**0.1399**	**0.8443**	**0.2254**
DIN	0.8649	0.2081	0.9169	0.1761	0.7644	0.2584
DG-DIN	**0.9283**	**0.1608**	**0.9644**	**0.1176**	**0.8331**	**0.2317**
FiGNN	0.8632	0.2121	0.9180	0.1753	0.7438	0.2635
DG-FiGNN	**0.9115**	**0.1767**	**0.9432**	**0.1501**	**0.8155**	**0.2406**

Table 4: Superiority of dual graph convolution.

Dataset	Alipay		Tmall		Alimama	
Model	AUC	Logloss	AUC	Logloss	AUC	Logloss
PNN	0.8756	0.2020	0.9261	0.1650	0.7758	0.2534
GCN-PNN	0.9036	0.1842	0.9402	0.1542	0.7953	0.2487
KGAT-PNN	0.9096	0.1796	0.9426	0.1510	0.7968	0.2467
HGAT-PNN	0.9119	0.1764	0.9433	0.1495	0.8002	0.2454
DG-PNN	**0.9216**	**0.1674**	**0.9501**	**0.1399**	**0.8443**	**0.2254**

Table 5: Effect of dual graph construction.

Dataset	Alipay		Tmall		Alimama	
Model	AUC	Logloss	AUC	Logloss	AUC	Logloss
PNN	0.8756	0.2020	0.9261	0.1650	0.7758	0.2534
attribute graph	0.9037	0.1831	0.9365	0.1545	0.8097	0.2428
uu & vv graph	0.9122	0.1753	0.9438	0.1473	0.8232	0.2353
uv graph	0.9109	0.1771	0.9437	0.1477	0.8221	0.2371
DG-ENN	**0.9216**	**0.1674**	**0.9501**	**0.1399**	**0.8443**	**0.2254**

optimizer [11]. In addition to the above hyper-parameters for all models, we tune the GCN layer size for graph models in the range of {1,2,3,4}. We use the validation set for tuning hyper-parameters, and the performance comparison is conducted on the testing set. We run each experiments 5 times and report the average results.

5.2 Performance Comparison

In this section, we compare the performance of DG-ENN with the state-of-the-art CTR prediction models. Table 2 shows the experimental results of all compared models on three datasets. We conduct Wilcoxon signed rank tests [4] to evaluate the statistical significance of DG-ENN with the best baseline algorithm. We have the following observations:

- DG-ENN consistently yields the best performance for all datasets. More precisely, DG-ENN outperforms the strongest baselines by **5.25%**, **2.59%** and **8.83%** in terms of *AUC* (*17.13%*, *15.21%* and *11.05%* in terms of *Logloss*) on Alipay, Tmall and Alimama, respectively. Possible reasons for the great improvement of DG-ENN over state-of-the-art CTR models may be the field-wise information propagation with attribute graph for alleviating the feature sparsity problem and the organized learning with user-item collaborative graph for behavior expanding across users and items. In contrast, most existing CTR methods ignore the rich relations existed in the data. We will further validate this observation in later experiments.
- LR performs worst among all baselines, which indicates that shallow linear combination of features is insufficient for CTR prediction. FM performs better than LR, proves that the effectiveness of second-order feature interactions. AutoInt+, DeepFM and PNN outperform FM, indicates that the modeling of high-order feature interactions is efficient for improving the performance of CTR prediction. DIN and DIEN achieve a comparable performance with DeepFM and PNN, demonstrates that user interest mining is also useful for representation learning.
- GIN applies graph convolution on item-item graph to enrich user behaviors. However, it ignores the rich attribute information and the complex relations between users and items, thus it behaves much worse than DG-ENN. FiGNN employs graph convolution on field graph to model feature interactions. As no other relation

information are introduced, it behaves no better than existing feature interaction based models.

5.3 Ablation Study of DG-ENN

In this section, we conduct a series of experiments to better understand the design rationality of our proposed DG-ENN.

5.3.1 On the compatibility of embedding enhancement. To investigate the compatibility of our proposed dual graph enhanced embedding, we integrate PNN, DIN and FiGNN with the dual graph enhanced embedding, which we named as DG-PNN, DG-DIN and DG-FiGNN. The experimental results are presented in Table 3. From these results, we can see that DG-PNN, DG-DIN and DG-FiGNN significantly outperform the original PNN, DIN and FiGNN models. It validates the compatibility of our embedding enhancement approach by demonstrating its effectiveness on working with various popular CTR models. This enhanced embedding is more informative with richer field-wise information and expanded user behaviors.

5.3.2 On the superiority of dual graph convolution. To demonstrate the superiority of our proposed dual graph convolution module, we consider the variants of DG-PNN with different graph convolution models on our constructed graphs. Specially, we compare dual graph convolution with GCN [12], KGAT [27] and HGAT [18]. Noticed that the original GCN, KGAT and HGAT are not designed for CTR prediction. We remove the prediction layer of these models and then apply them on our constructed graphs for embedding

enhancement. We named these variants as GCN-PNN, KGAT-PNN and HGAT-PNN. Table 4 summarizes the results, from which we have the following findings:

- All these embedding enhanced models outperform the original PNN model, further verifies the effectiveness of embedding enhancement with relational information represented as graph.
- KGAT-PNN behaves better than GCN-PNN on all three datasets. A possible reason is that GCN models the constructed graphs as a homogeneous graph, which ignores the different chasracteristics of differessnt fields while KGAT considers such differences.
- HGAT-PNN outperforms KGAT-PNN on all three datasets. This is because that HGAT-PNN utilizes all the graphs and models them in an heterogeneous manner, while KGAT only considers the collaborative graph and item-attribute graph.
- DG-PNN consistently outperforms all baselines, which validates the superiority of our proposed dual graph convolution.

5.3.3 On the effect of dual graph construction. We conduct experiments on three datasets to validate the effectiveness of the construction of attribute graph and collaborative graph. We divide the collaborative graph into two parts: (1) user-user edges combined with item-item edges and (2) user-item edges, for detailed comparison. Specially, we design four comparing variants: (1) DG-ENN only with the attribute graph (named attribute graph), (2) DG-ENN only with the user-user edges and item-item edges in the collaborative graph (named uu & vv graph), (3) DG-ENN only with the user-item edges in the collaborative graph (named uv graph) and (4) DG-ENN with neither attribute graph nor collaborative graph (that is PNN). Table 5 shows the comparison between different variants. We observe that PNN performs the worst in all these models, which proves the effectiveness of attribute graph and collaborative graph. Moreover, we find that DG-ENN performs better than all the other models. It indicates that these attribute graph and collaborative graph are complementary to each other and can be combined together to improve the embedding quality and therefore boost the model performance.

5.4 In-depth Analysis on Graph Modeling

5.4.1 Impact of Aggregators. To explore the impact of different aggregators, as formulated in Equation 9-11, we compare the performance of our proposed model with different aggregators. Figure 3 summarizes the experimental results. We can see that GCN aggregator performs better than NGCF aggregator on all datasets. A possible reason is that additional feature interactions between central node and neighbor nodes introduced by NGCF aggregator makes it easy to overfit. Moreover, we can see that LightGCN aggregator which removes both the weight matrix and activation function achieves the best performance on all datasets.
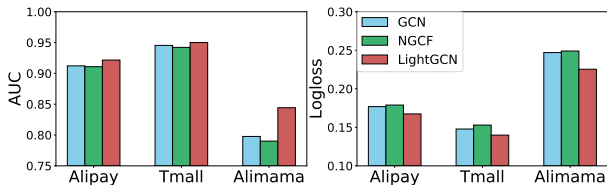

Figure 3: Impact of Aggregators.

5.4.2 Impacts of Attribute Information Exploitation. To verify the effectiveness of our divide-and-conquer strategy to integrate different attribute information, as explained in Section 4.2, we replace our proposed attribute graph convolution module with other two alternatives: (1) using the linear transformation of ID embedding and attribute embeddings as the refined user/item representation [2, 12], (2) modeling the different fields of attributes without considering their fields. Figure 4 shows the experimental results, we can see that the first alternative gets the worst performance, proving the effectiveness of modeling attributes as graphs. Besides, modeling the different fields of attributes without considering their fields (i.e., the second alternative) performs worse than our model, verifies the necessary of modeling field-wise information individually.
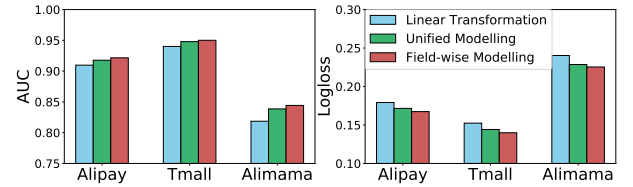

Figure 4: Impact of Attribute Information Exploitation.

5.4.3 Impacts of Collaborative Signal Exploiting. To validate the superiority of our design of organized learning for the collaborative graph, as explained in Section 4.3. We conduct three different operations on the aggregated embeddings from multiple types of edges: (1) element-sum operation; (2) element-mean operation; (3) attention operation. From the results in Figure 5, we can see that our DG-ENN obtains the best results. Besides, we find that attention operation achieves the second best results.
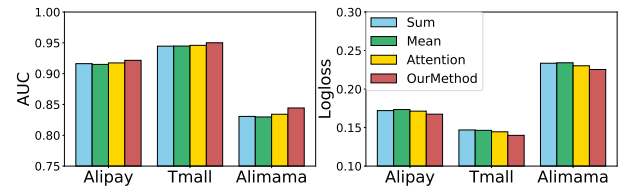

Figure 5: Impacts of Collaborative Signal Exploiting.

5.5 Case Study

In this part, we conduct experiments to verify that our model can solve the problem of feature sparsity and behavior sparsity.

5.5.1 Feature Sparsity Analysis. In order to prove that our model can solve the feature sparsity well, we select instances in the test set containing one of the four features with low frequency in the training set. The four chosen features are presented in Table 6, where they are represented by feature fields with subscripts of desensitization information. We report the performance (i.e., Logloss) of PNN and DG-PNN on the selected test instances in Table 6. We can find that DG-PNN achieves significant performance improvement on the test samples with sparse features, compared to PNN. This result demonstrates that our proposed dual graph enhanced embedding alleviates the feature sparsity issue.

Table 6: Feature Sparsity Analysis in Alimama.

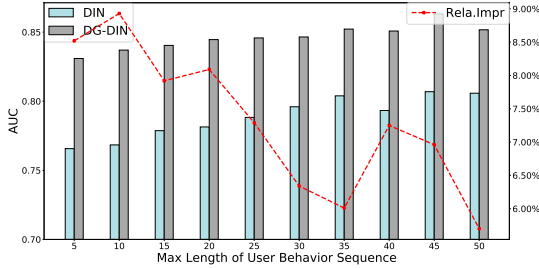| Feature	Frequency	PNN (Logloss)	DG-PNN (Logloss)
Brand_1	12	0.3502	0.2868
Brand_2	5	0.3218	0.3111
Cate_1	8	0.6125	0.5645
Cate_2	9	0.0851	0.0223

Figure 6: Behavior Sparsity Analysis.

5.5.2 Behavior Sparsity Analysis. Besides, the behavior sparsity problem can also be solved well by our model. We choose Alipay dataset for experiment because this dataset includes less attribute information which may make noise for behavior sparsity analysis. Figure 6 shows the performance comparison between DIN and DG-DIN with respect to different lengths of user behavior sequences. The result shows that the relative improvement of DG-DIN over DIN is more significant when length of user behavior sequence is less. That is to say, our proposed dual graph enhanced embedding alleviates the behavior sparsity issue.

6 CONCLUSIONS

In this paper, we focus on exploiting the graph representation learning to alleviate the feature sparsity and behavior sparsity problems for existing CTR models. We propose a novel dual graph enhanced neural network based on attribute graph and collaborative graph. On the one hand, to learn the feature representation from attribute graph effectively, we propose a divide-and-conquer learning strategy to perform field-wise attribute modeling. On the other hand, to model the complex user-item relation for behavior expanding, we design a organized learning strategy inspired by curriculum-learning to learn the correlations within users/items and also between users and items. The extensive experiments on three real-world datasets have demonstrated the superiority of our proposed DG-ENN over the state-of-the-art methods. Moreover, the proposed dual graph enhanced embedding is able to work collaboratively with various deep CTR models to boost their performance.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

[2] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph convolutional matrix completion. In *SIGKDD Workshop on Deep Learning Day*.

[3] Heng Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. (2016).

[4] Ben Derrick and Paul White. 2017. Comparing two samples from an individual Likert question. *International Journal of Mathematics and Statistics* 18, 3 (2017).

[5] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *International Joint Conference on Artificial Intelligence*. 1725–1731.

[7] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*. ACM, 355–364.

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.

[9] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*. 1531–1540.

[10] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 43–50.

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[13] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past erformance data. In *SIGKDD*. ACM, 768–776.

[14] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *CIKM*. 2615–2623.

[15] Feng Li, Zhenrui Chen, Pengjie Wang, Yi Ren, Di Zhang, and Xiaoyu Zhu. 2019. Graph Intention Network for Click-through Rate Prediction in Sponsored Search. In *SIGIR*. 961–964.

[16] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *CIKM*. 539–548.

[17] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. *arXiv preprint arXiv:1803.05170* (2018).

[18] Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP-IJCNLP*. 4823–4832.

[19] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted Factorization Machines for Click-Through Rate Prediction in Display Advertising. In *WWW*. 1349–1357.

[20] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. In *SIGIR*. 2347–2356.

[21] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-based Neural Networks for User Response Prediction over Multi-field Categorical Data. *arXiv preprint arXiv:1807.00311* (2018).

[22] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong sequential modeling with personalized memorization for user response prediction. In *SIGIR*. 565–574.

[23] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.

[24] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[25] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*. 1161–1170.

[26] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. ACM, 12.

[27] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *SIGKDD*. 950–958.

[28] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[29] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[30] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[31] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*. 974–983.

[32] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *SIGKDD*. 793–803.

[33] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.

[34] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *SIGKDD*. ACM, 1059–1068.