IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 April 2023, accepted 17 April 2023, date of publication 20 April 2023, date of current version 3 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3268616

RESEARCH ARTICLE

IcaGCN: Model Intents via Coactivated Graph Convolution Network for Recommendation

JINGXUE ZHANG[1] **AND CHANGCHUN YANG**[2]
[1]School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou 213164, China
[2]School of Microelectronics and Control Engineering, Changzhou University, Changzhou 213164, China

Corresponding author: Changchun Yang (ycc@cczu.edu.cn)

This work was supported in part by the Jiangsu Province Graduate Research Innovation Program.

ABSTRACT In this era of information overload, to better provide personalized content services to users, recommendation systems have greatly improved the efficiency of information distribution. Graph Convolution Network(GCN), which is one of the representative works of graph structure aggregation processing, works by node convolution with the help of the Laplacian matrix of the graph and weighted combination of neighbor node information according to the outgoing and incoming degrees of neighbor nodes to obtain the representation of the current node. However, the mainstream GCN models nowadays do not take into account data augmentation of metadata and the fact that each node plays different roles with different importance and weights, thus making the recommendation performance limited. To better solve the above problems, we propose the IcaGCN model, which can perform data augmentation and calculate node weights in modules, and is a convenient plug-and-play method. Finally, extensive experimental results on four real-world datasets have shown the effectiveness and robustness of the proposed model. Especially on the Amazon-Book dataset, our IcaGCN has improved by 6.32%, 42.29%, and 12.38% in Recall@20, MRR@20, and NDCG@20, respectively, compared to other existing state-of-the-art models. We also provide source code and data to reproduce the experimental results available at https://github.com/PersonZ1223/IcaGCN.git

INDEX TERMS Recommender systems, graph neural networks, collaborative filtering.

I. INTRODUCTION

In this era of information overload, in order to better provide personalized content services to users, the birth of recommendation systems has greatly improved the efficiency of information distribution and gradually become one of the infrastructures of information services. Reviewing the development history of recommendation system, it can be divided into traditional shallow model stage, general neural network model stage and graph neural network model stage [3]. The essence of recommendation system is to solve the matching problem between users and items, i.e., to learn the similarity between users and items. Early collaborative filtering (CF) was done by capturing the similarity of users/selected items, however, the disadvantage of such shallow models is that it is difficult to make full use of data in the face of complex user behavior, which leads to unsatisfactory recommendation results [12]. The rise of deep learning research, however, has made it possible for machines to process large-scale complex data, and neural network-based models have gradually replaced shallow models. The multi-layer perceptron (MLP) [8] in neural collaborative filtering models replaces the dot product operation of matrix factorization (MF) [13], which is the most classical of collaborative filtering methods. However, both of the above models deal with Euclidean spatial data, and they both ignore the structured information of the data. That is, they generally utilize only the information of first-order neighbor nodes on the graph, leading to limited recommendation performance.

In the real world, data has more complex relationships, and the use of "graph" to represent data becomes a new solution. Take social network as an example, the graph is constructed with users as nodes and relationships as edges, each node has no fixed number of neighbors, and there is

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

no clear sequential relationship between nodes. The graph neural network (GNN) is based on "graph" structure data, and inferring and reasoning by deep learning the relationship information in the graph, the complex data in non-Euclidean space can be represented as a graph. This message propagation mechanism allows nodes to aggregate information from higher-order neighbors, thus effectively capturing higher-order relationships for more effective representation learning. Therefore, graph neural networks have stronger learning ability and are expected to solve relational inference problems and improve interpretability. Among them, graph convolutional network (GCN), as one of the representative works of graph structure aggregation processing, works by node convolution with the help of Laplacian matrix of the graph and weighted combination of neighbor node information according to the outgoing and incoming degrees of neighbor nodes to obtain the representation of the current node.

The message propagation mechanism in the graph convolutional network should make a distinction for the importance of the user's intentions to different items. Different items have different influence on user interest learning, so the different weights of different nodes should be taken into account when propagating. As shown in the FIGURE 1, the user on the left chooses to buy supplies related to his occupation when shopping, but these supplies are not suitable to be recommended to most people. But the user on the right has social influence, so what he buys also becomes something that many people like, so the user on the right should have more weight than the user on the left. Similarly, if an item is more niche, such as the diamond ring shown in the FIGURE 1, then it has less weight because of the small audience. In addition, data augmentation is also important to ensure that user and item metadata are not ignored. Therefore, our proposed IcaGCN model can process the data from two aspects, data augmentation on the one hand, and assigning different weights to different nodes on the other.

To summarize, this work makes the following main contributions:

- We propose the IcaGCN model, in which the Coactivate Unit can augment user data and item data, and the Cube Unit can assign different weights to different nodes.
- IcaGCN model obtains better results than other baseline models. Especially on the Amazon-Books dataset, our IcaGCN has improved by 6.32%, 42.29%, and 12.38% in Recall@20, MRR@20, and NDCG@20, respectively, compared to LightGCN.

II. PRELIMINARIES

In this section, we will briefly explain several methods for deriving node weights led by the attention mechanism, and discuss several currently available methods that can combine user and item raw data and embeddings to maximize the use of user embeddings and item embeddings without losing important information. In addition, we will explain in detail the baseline model of IcaGCN: LightGCN.

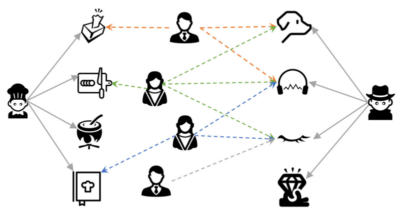

FIGURE 1. Different users and items in shopping sites.

A. TYPICAL ATTENTION MECHANISM

The attention mechanism originated from the fact that humans can naturally and effectively discover salient regions in complex scenes, and then with the development of computer technology, this attention mechanism can be regarded as a dynamic weight adjustment process based on input features [6]. The way to combine deep neural networks with attention mechanisms starts in the Residual Attention Module(RAM) [23] by repeatedly predicting important regions and updating the entire network in an end-to-end fashion. Subsequently, research in this area transitioned to explicitly predicting the input features to be discriminated, of which Deep Crossing Network(DCN) [24] is an important example. After the channel attention network represented by the Squeeze-Excitation network (SENET), [9] is generated, the model can adaptively predict potential key features. GCN assigns the same weight to different neighbors in the same-order domain, which limits the capture of spatial information correlation, and then GAT [21] based on the attention mechanism weighted summation of neighbor node features, according to different neighbor node features assign different weights to it.

B. CHANNEL ATTENTION MECHANISM

Hu et al. [9] proposed the SENET which learns the importance of each feature channel is automatically obtained, and then useful features are promoted and unuseful features for the current task are suppressed according to this importance, so the interdependence between feature channels is explicitly modeled relationship to enhance the directivity of the extracted features. There are many applications of SENET, such as skin lesion classification, recalibrating fully convolutional networks and context aggregation. Dai et al. [2] proposed a multi-scale channel attention module(MS-CAM) which adds the global context inside the attention module. Given the global channel context $g(X)$, the $g(X)$ can be obtained as follow:

$$g(X) = \mathrm{GP}(\mathrm{BN}(\mathrm{PWConv}(\delta(\mathrm{BN}(\mathrm{PWConv}(X)))))). \quad (1)$$

BN denotes batchnorm, PWConv denotes point-wise channel interactions for each spatial position, δ denotes relu and GP denotes global average pooling. The refined feature $X' \in \mathbb{R}^{C\times H\times W}$ by MS-CAM can be obtained as follow, in which σ denotes sigmoid:

$$X' = X \otimes \sigma(g(X)). \quad (2)$$

C. FEATURE INTERACTION

Feature interaction plays an important role in recommender systems. Feature intersection refers to the cross combination between two and more original features, and how to mine higher-order feature interactions is one of the key directions of current research. Liu et al. [14] uses CNN architecture combined with pooling layer and MLP layer to form a higher-order feature interaction extraction structure, and uses this part of features as input together with original features into the subsequent network structure to get the prediction of samples. CNN as a classical method for processing image data is incorporated into the recommendation system. Liu et al. [16] introduces a weight parameter for each feature interaction out of the idea that the interaction between individual features is not equally important, thus reflecting the importance of that feature interaction. Liu et al. [15] makes the model automatically mine the interaction features by setting different feature grouping, and each group is responsible for mining different order of feature interaction information, which means that AutoGroup shows not only 2nd order but even arbitrary higher order interactions when mining feature interactions. All the above examples illustrate the positive effect of mining effective higher-order feature interactions on the model. However, most people tend to characterize users and items as vectors for learning, i.e., features are non-linearly and implicitly transformed into embeddings, ignoring the interaction between the original data of users and items. In contrast, the Cartesian product, an excellent method for data interaction, has the problem of a huge number of parameters.

D. GCN AND LightGCN

Euclidean spatial data with regular data and fixed shape, such as image data, are mostly processed by DNNs. To solve the data in non-Euclidean space, GNN and GCN are born. Data in non-Euclidean space is widely existed in scenarios such as social networks, virus propagation paths, food chains, particle networks, etc. The data in these scenarios are best represented by graphs. GNN takes graphs as input and gets prediction results as output. In a graph neural network, the information of the current node and its n-hop neighbors nodes are aggregated and updated. For GNN, a graph network needs the input of node feature matrix and adjacency matrix so that the aggregation operation of nodes can be performed. In a graph convolutional network, the weighting of the adjacency matrix itself is taken into account and the rows and columns of the degree matrix are normalized such that the embedding of the current node in the current layer is equal to the weighted sum of its neighboring nodes as well as itself. Thus, the computation of each layer in a graph convolutional network can be viewed as matrix multiplication. Briefly, GNN aggregates the nearest neighbor information and GCN aggregates the edge weight information [19]. GCN, which learns node representation by smoothing features on the graph, iteratively performs the neighbor aggregation operation of graph convolution as follows:

$$e_u^{(k+1)} = AGG(e_u^{(k)},\ e_i^{(k)} : i \in N_u\}). \tag{3}$$

NGCF [25] divides the propagation process into two parts: message construction and message aggregation. The model also retains the two most common designs in GCN: feature transformation and nonlinear activation. The user and item embeddings predicted by the LightGCN model [7] through iterative and weighted summation are as follows:

$$e_u^{(l+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_i^{(l)}, \tag{4}$$

$$e_i^{(l+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e_u^{(l)}. \tag{5}$$

Each layer of user embeddings and item embeddings are involved in a multilayer graph convolutional neural network and are summed according to the weights. In the authors' experiments, it is found that learning that weight with average weight gives better results. The weighting formula is as follows:

$$e_u = \sum_{k=0}^{K} \alpha_k e_u^k,\ e_i = \sum_{k=0}^{K} \alpha_k e_i^k. \tag{6}$$

III. OUR MODEL

In this section, we first give an overview of our proposed method IcaGCN to briefly describe how it computes the weights of each node while preserving the user embedding and item embedding. Then, we will explain in detail the principle, the role of each step, and the shape change of the data in a hierarchical manner according to the flow of the data.

A. OVERVIEW

The goal of our method is to preserve the impact of user embeddings and item embeddings themselves as much as possible, and to derive the weight of each node by a non-attention mechanism method, so that the impact of important nodes is greater than the impact of non-important nodes. As shown in FIGURE 2, after first processing the user and item metadata to obtain the user embeddings and item embeddings, the user embeddings and item embeddings are stitched together into the Coactivate Unit, which serves as a data enhancement. Then, it enters the Cube Unit, where the weights of individual nodes can be extracted relative to the global one. When the data flows out of the IcaGCN Layer, the weights of each node will be obtained, and the weights will be brought to the adjacency matrix through L-layer information transfer, so that each node plays a different role: the role of important nodes is more pivotal.

Once the nodes that have obtained the weights have gone through the L-layer information transfer, they follow the LightGCN approach to train the model with the traditional transpose of the user embeddings and the Hadamard product of the item embeddings as the prediction result. In this way, the IcaGCN model is able to distinguish between significant and non-significant nodes while preserving the influence of user embeddings and item embeddings, thus optimizing the prediction results. In the following, we elaborate more details of the method we propose.

B. INPUT LAYER

Out of concern for the overly sparse nature of the data set present, we first represent the user attributes and item attributes with sparse vectors. Then, we use one-hot coding, which is computationally convenient and fast and expressive, to process the data. At the same time, in order to solve the drawback of one-hot encoding for too sparse data, which has the disadvantage of over-utilizing resources, we introduce the embedding layer for dimensionality reduction operation. We keep the operation of this layer consistent with Light-GCN, and assume that the number of users is M and the number of items is N. Then the obtained user embeddings and item embeddings can be expressed as:

$$E_u = [e_{u_1}, \ldots, e_{u_M}],\ E_i = [e_{i_1}, \ldots, e_{i_N}]. \tag{7}$$

where D denotes the embedding size, $\boldsymbol{e}_u \in \mathbb{R}^D$ denotes user embeddings while $\boldsymbol{e}_i \in \mathbb{R}^D$ denotes item embeddings. We input the embedding obtained after splicing E_u and E_i to the next layer. That is, the input of IcaGCN Layer can be expressed as:

$$E = [e_{u_1}, \ldots, e_{u_M}, e_{i_1}, \ldots, e_{i_N}]. \tag{8}$$

C. IcaGCN LAYER

For $E = e_u \oplus e_i$, the size of this embedding is $[N, D]$. Next, we have to go through the Coactivate Unit and the Cube Unit in turn, where the former serves for data augmentation and the latter serves for computing the weight of each node. In the following two subsections, we will explain in detail.

1) COACTIVATE UNIT

In the graph-based feature interaction method, GCN is the aggregation of information from edge weights. In the actual recommendation projects, it is easy to produce the situation that the original data is ignored, which leads to the degradation of the recommendation effect. Therefore, it is necessary to enhance the data before processing them. In Coactivate Unit, we set a variable parameter order, which can be set by ourselves. Regarding the effect of the size of this parameter on the model results, ablation experiments are performed in Section IV of this paper. The significance of order is to specify the degree of enhancement of the original input and the number of fully connected layers. The following equation shows the mathematical form of the coactivate unit:

$$order = i + 1; \tag{9}$$

$$h_0 = E = [e_u \oplus e_i]; \tag{10}$$

$$h_i = w_{i-1} \otimes h_{i-1} + b_{i-1}; \tag{11}$$

$$E_{coa} = \sum_{i=1}^{order-1} h_i. \tag{12}$$

where order is set to $i+1$ and the original input E is assigned to h_0. The weights and bias terms of each fully connected layer are set by random initialization of the Gaussian distribution. The final output is obtained from the cumulative h_i, so the Coactivate Unit does not change the shape of the input and remains $[N, D]$ unchanged.

Our proposed coactivate unit has several advantages over other methods: First, we take full advantage of the computational advantages of MLP to fully exploit the potential of the original input and perform this task of data augmentation well. Second, in terms of time complexity, the approach using fully connected layers is more space and time efficient than the approach using Cartesian products. Fewer parameters also mean less time complexity. The traditional cartesian product to solve similar problems requires a time complexity of $O(N^2 \times D)$, while the time complexity of our proposed unit is only $O(N \times D)$, where D represents the dimension of embedding and N represents the total number of users and items. Third, the coactivate unit is very convenient and plug-and-play. In Section IV of this paper, we show the model code, and we can see that our proposed method can be applied not only to graph convolutional networks, but also to other network models.

2) CUBE UNIT

From the MS-CAM method mentioned in the previous section, we consider whether there exists a non-attention mechanism plug-and-play method that can, like the attention mechanism, serve to make different nodes play different roles with different importance, i.e., important nodes have more weight and non-important nodes have less weight. In dealing with the image problem, the MS-CAM method uses global average pooling (GAP) to make the image reduced from 3 dimensions to 1 dimension, outputting a response operation for each feature map. After that, the convolution layer and regularization operation are used to find the weights of different regions of the image. In analogy to the recommendation class problem, we first consider the use of linear layers instead of convolutional layers, which greatly reduces the number of parameters and increases the training speed. Finally, we use the relu activation function and dropout operation instead of the original regularization operation to enhance the plug-and-play nature. The following formulas, executed sequentially, represent the mathematical form of the Cube Unit:

$$Cube_0 = E = [e_u \oplus e_i]; \tag{13}$$

$$Cube_1 = Global\_max\_pooling(E); \tag{14}$$

$$Cube_2 = max(0, w \otimes Cube_1 + b); \tag{15}$$

$$Cube_3 = dropout(Cube_2); \tag{16}$$

$$E_{cube} = \frac{1}{1 + e^{-Cube_3}}. \tag{17}$$

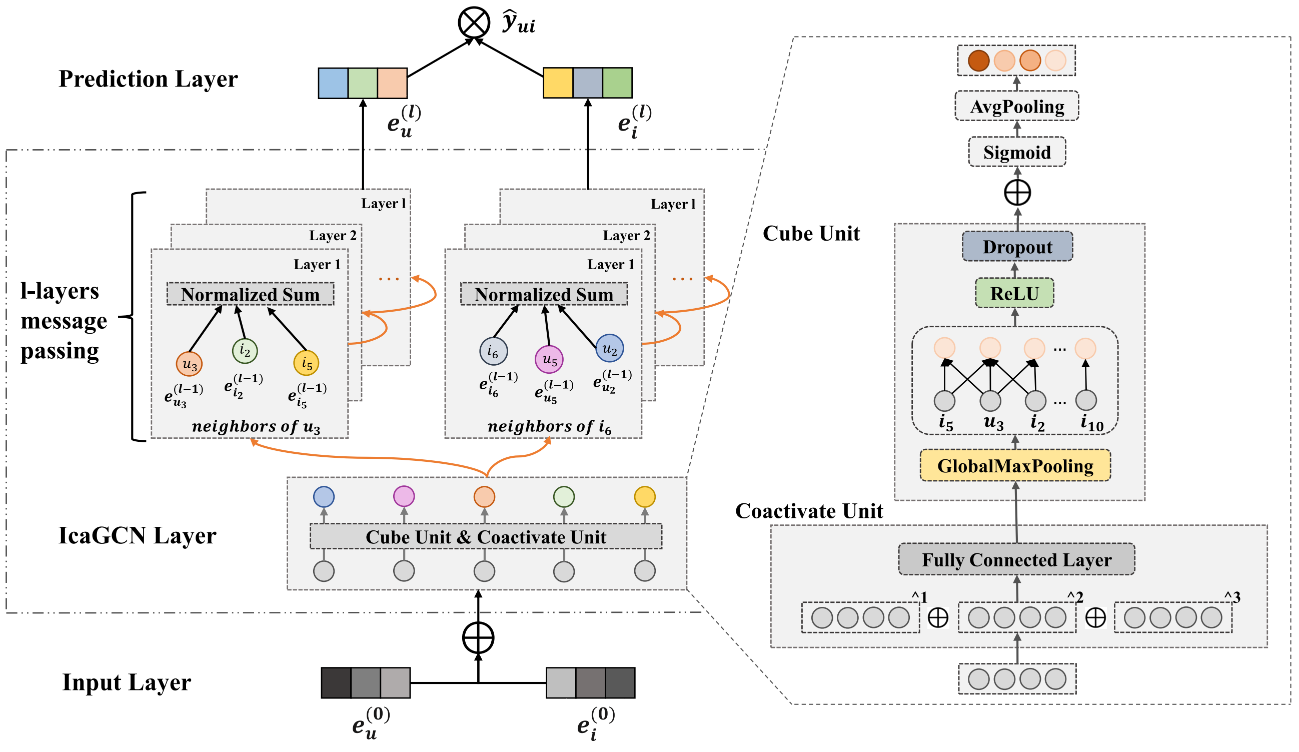

FIGURE 2. Illustration of the IcaGCN model.

after the embedding with shape $[N, D]$ enters the Cube Unit, it first undergoes global max pooling for dimensionality reduction, and the shape becomes $[N, 1]$. This step of the operation can transform the shape more naturally without using a large number of parameters. After straightening the embedding, it enters the linear layer and continues the computation of the activation function and randomly discards some nodes according to the situation. Regarding the rounding probability parameter in the dropout operation, we will do ablation experiments later for comparison. After the data flow out of the Cube Unit, we limit the size of the obtained weight scores to between [0, 1] by sigmoid operation and weight the user embedding and item embedding.

The advantages of Cube Unit are numerous: first, it follows that the attention mechanism mentioned in LightGCN may not be able to play a more active role in such a simple model. Second, it improves the approach of the image domain and applies it reasonably well to the recommendation domain, plug-and-play computing the weights of each node. Third, it is efficient and convenient. While ordinary attention mechanisms need to recalculate node weights in each round, this method only needs to calculate them once, bringing advantages such as a small number of parameters and fast training, and its time complexity is only $O(N \times D)$.

D. PREDICTION LAYER

After passing through the IcaGCN layer, we obtain the weighted user embeddings and item embeddings. using the same L-layer messaging strategy as LightGCN, the data enters the prediction layer. We use the weighted $e_u^{(0)}$ to represent all users and $e_i^{(0)}$ represents all the items. We get a final representation of the user and item via L-layers normalized sum:

$$e_u = \sum_{l=0}^{L} a_l e_u^{(l)};\ e_i = \sum_{l=0}^{L} a_l e_i^{(l)}. \tag{18}$$

where α_l means the importance of l-th layer, which keeps the same setting with LightGCN [7]. We use the inner product of user and item final representations to define the model prediction:

$$\hat{y}_{ui} = e_u^T e_i. \tag{19}$$

E. MODEL TRAINING

For this processing step, we choose the same way as LightGCN: Bayesian personalized ranking (BPR) [20] loss to calculate the loss of the model. BPR loss is pairwise loss, for which the predicted value of entries that can be observed will be higher than the predicted value of entries that cannot be observed. It does ranking optimization for each user's own item preference decibel, and introduces a Bayesian prior, thus reducing the overfitting of the model. The formula is shown as follows:

$$Loss_{BPR} = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \parallel E^{(0)} \parallel^2 . \tag{20}$$

TABLE 1. Statistic of the datasets.

Dataset	#User	#Item	#Interact	Sparsity
Movielens-1M	6040	3629	836478	96.18%
Gowalla	29859	40989	1027464	99.92%
Yelp2018	45478	30709	1777765	99.87%
Amazon-Book	35737	38122	1960674	99.86%

where λ controls the strength of $L2$ regularization. In addition, we use the Adam optimizer in small batches to help with training, which has the advantages of fast gradient descent and ease of use. Since the focus of this model is on the update of the network structure, model training is not the focus of our work, and we leave the optimization strategies such as the optimal loss function for future study.

IV. EXPERIMENTS

In this section, we specifically present several empirical studies of the IcaGCN model. In Sec.A, we describe the four datasets used in the experiments. In Sec.B, we detail the experiments' settings. In Sec.C, to better validate the IcaGCN model, we compare it with other state-of-the-art models on four real and large datasets; In Sec.D, to specifically discuss the influence of each module in the IcaGCN model on the experimental results, we modify the values of some parameters in the module and learn extensively how the IcaGCN model affect recommendations. In addition, we designed more ablation experiments to demonstrate the effectiveness of the model by comparing it with existing data-enhancement models, adding contrast learning to IcaGCN for further improvement, and finally comparing the training efficiency.

A. DATASET DESCRIPTION

To better show the effectiveness of our IcaGCN model, we use four datasets: Movielens-1M, Yelp2018, Gowalla and Amazon-Book, which are large and sparsity, and span a wide range of domains. We show the statistics of four datasets in Table 1.

- Movielens-1M is a movie rating dataset: it includes movie ratings, movie metadata (genre type, era) and demographic data about users (age, zip code, gender, occupation, etc.). Among them, our experiments mainly use *user_id* and *item_id* to construct a bipartite graph of user items, and user scores above 3 are considered as valid data.
- Yelp2018 is a publicly available dataset from Yelp, the largest review site in the United States. The dataset contains customer reviews of restaurants, bars, and other business establishments. Among them, our experiments mainly use *user_id* and *business_id* to construct a bipartite graph of user items, and ensure that there are at least 15 interactions between them.
- Gowalla is a check-in dataset from the United States. Each check-in record for each user is represented as one record with the specific attributes of the check-in broken out. Each check-in record includes *user_id*, check-in time, latitude of check-in location, longitude of check-in location, and *location_id* that uniquely corresponds to the latitude and longitude of each location. where our experiments mainly use *user_id* and *location_id* to construct a bipartite graph of user items, and ensure that there are at least 10 interactions between them.
- Amazon-Book provided by Amazon. The dataset includes Amazon's product reviews and metadata. Among them, our experiments mainly use *user_id* and *item_id* to construct a bipartite graph of users-items, and ensure that there are at least 15 interactions between them.

For each user-item interactions, we treat it as a positive instance, and then conduct the negative sampling strategy to pair it with one negative item that the user didn't consume before.

B. EXPERIMENTAL SETTINGS

1) EVALUATION METRICS

For each user in the test set, we specify all items that the user has not interacted with as negative items. To better validate the effectiveness of the IcaGCN model applied to top-k recommender systems, we adopt three widely used metrics: recall@k, mrr@k, and ndcg@k. We set k=20 by default. In the following, we briefly describe the implications of these three criteria.

- recall@k The ratio of the number of relevant results retrieved from the top-k results to the number of all relevant results in the library, and measures the search completion rate of the retrieval system [17]. The larger this value is, the better.
- mrr@k Mean reciprocal rank [18], an internationally used mechanism for evaluating search algorithms, using the ranking of the correct search result value in the search results to assess the performance of the retrieval system. The larger this value is, the better.
- ndcg@k Normalized discounted cumulative gain [26], an evaluation metric that takes into account the order of return. The larger the effect the better.

2) BASELINES

We use recbole [33] as the codebase for experiments. In the following, we briefly describe the baselines we compared with.

- **BPRMF [11]** This is a basic algorithm in the field of recommendation systems. The collaborative filtering algorithm mainly predicts users' ratings of items and thus makes recommendations, but the BPRMF algorithm sorts each user's items of interest by preference, and this approach is very popular for handling implicit feedback data.
- **NeuMF [8]** It combines generalized matrix factorization(GMF) and multi-layer perceptron(MLP). It can extract both low and high dimensional features. The

GMF module converts user_id and item_id into user feature vectors and item feature vectors, while the MLP module focuses on learning nonlinear relationships.

- **GCMC [1]** This is a message passing based graph self-coding framework that considers both side information and network structure, and analyzes the impact of side information in cold starts of recommender systems. The main contribution is the application of GNN to matrix-completion tasks with side information.
- **GAT [22]** An attention mechanism is introduced to assign different weights to each node, learn node features and structural features of nodes in the graph, and process local information while paying attention to the overall information. The core idea is to assign attention only to the first-order neighbors of the nodes.
- **NGCF [25]** It enhances embedding by explicitly modeling the high-order connectivity between user-item, and mining the higher-order connectivity relations to capture interactions to refine the multiple embedding propagation of embedding. It is deformed from the standard GCN using a nonlinear activation function and feature transformation matrix.
- **LightGCN [7]** It removes the nonlinear activation function and feature transformation matrix from NGCF, and only adds a set of weight coefficients to neighborhood aggregation of embeddings output from different GCN layers as the final embedding, and only aggregates neighbor nodes, thus greatly simplifying the model.

3) PARAMETER SETTINGS

We use PyTorch to implement all models in our experiments. The GPU of the server we use is P100-16G and the CPU is Intel(R) Xeon(R) CPU E5-2690 v4. The python version is 3.8, the cuda version is 10.2, and the torch version is 1.11.0. For the embedding layer and optimization method, we use $\mathcal{L}_2$ regularization with 10^{-4} weight, the learning rate is set to 10^{-3}, the batch size of training to 4096 and batch size of evaluating to 204800, the negative sampling radio $\mathcal{R}$ to 300, we fix the embedding size to 64 and this is same to many recent GCN-based models like LightGCN. For all deep models, the depth of layers is set to 3 and we use RELU as an activation function and the dropout rate is set to 0.5.

C. PERFORMANCE COMPARSION

We first report the performance of our method in Table 2, and then we have the following observations:

- IcaGCN has the best performance in all the four datasets mentioned above. Among them, on the Amazon-Book dataset, IcaGCN improves over LightGCN in recall@20, mrr@20 and ndcg@20 by 6.32%, 42.29% and 12.38%, respectively. This also shows the effectiveness of the proposed method of IcaGCN to combine the information of user embedding and item embedding itself, and then find the weight of each node. Summarizing the excellent results shown by our model, we attribute it to the following reasons: 1) Compared with previous GCNs, IcaGCN further widens the gap between the different roles played by different nodes without losing the information of the users and the item itself in the process of exploring higher-order connectivity, which also indicates that the method used for image processing has certain interoperability with the method used for recommender systems; 2) Compared with other baselines, IcaGCN can effectively utilize the importance of nodes and learn them. These advantages together lead to the superiority of IcaGCN to outperform SOTA models.
- We believe that the results of the model are related to the amount of data in the dataset. the Gowalla dataset and the Yelp2018 dataset have comparable amounts of data, and the gains of IcaGCN on these two datasets are not very different. The Amazon-Book dataset has the largest and sparsest data volume, and our model achieves better results on this dataset, especially the gain of mrr@20 illustrates that the improved IcaGCN model advances almost half of the ordinal number of the first item cited in the top-20 recommendations that meets users' expectations.
- In general, the graph convolutional network model outperforms the traditional web embedding model, but this advantage is not obvious on the Movielens-1M dataset. The reason for this is that the Movielens-1M dataset has a small amount of data, while the graph convolutional approach is more suitable for a cluttered and large amount of dataset. The graph convolutional network is better at mining collaborative information and is more suitable for complex datasets than random wandering and heuristic mining strategies. All technical improvements work because they fit the characteristics of our data, and the data support the strengths of the model. For sparse and simple datasets, the structure of the model often does not need to be too complicated, and the success of LightGCN also illustrates this.

D. STUDY OF MODEL

In order to deeply explore the role played by each module of IcaGCN, we designed some ablation experiments for validation: 1) we analyzed the effect of order on the results, and designed the range of order values among [1,2,3,4]; 2) we experimented with the number of Cube Unit values, and set them among [1,2,3,4]; 3) the dropout parameter in the Cube Unit is discussed to explore the relationship between its size and the dataset; 4) comparison with existing data enhancement methods; 5) comparison of single-round training time; 6) utilize the contrastive loss to learn user and item embeddings and compared to traditional collaborative filtering methods.

1) EFFECT OF ORDER

The parameter order reflects to some extent the degree of input of the original information of user and item. When order=1, the user and item embeddings themselves go

TABLE 2. Overall performance comparison. IcaGCN denotes our model when dropout ratio=0.5, the number of Cube is 1 and order=1. R, M, and N denote recall, mrr, and ndcg, respectively. Improv. denotes the relative improvements over the best GNN-based baselines. The results of significance testing indicate that our improvements over the current strong GCN-based baseline are statistically significant(p-value<0.05).

Metrics		BPRMF	NeuMF	GCMC	GAT	NGCF	LightGCN	IcaGCN	Improv.
Amazon-Book	R@20	0.0968	0.0655	0.0908	0.0921	0.0937	0.1123	**0.1194**	6.32%
	M@20	0.0678	0.0389	0.5860	0.6171	0.0629	0.0778	**0.1107**	42.29%
	N@20	0.0389	0.0336	0.0557	0.0574	0.0580	0.0638	**0.0717**	12.38%
Gowalla	R@20	0.1517	0.1447	0.1497	0.1514	0.1572	0.1898	**0.2056**	6.32%
	M@20	0.1114	0.0901	0.0904	0.1538	0.1074	0.1307	**0.1442**	10.33%
	N@20	0.1001	0.7950	0.0807	0.0866	0.0904	0.1106	**0.1238**	11.93%
Yelp2018	R@20	0.1000	0.7610	0.0924	0.0939	0.0963	0.1134	**0.1185**	4.50%
	M@20	0.0690	0.0466	0.0601	0.0625	0.0644	0.0772	**0.0811**	5.05%
	N@20	0.0557	0.0396	0.0495	0.0514	0.0523	0.0628	**0.0656**	4.46%
Movielens-1M	R@20	0.2665	0.2483	0.2630	0.2691	0.2725	0.2677	**0.2852**	6.54%
	M@20	0.4248	0.3906	0.4232	0.4211	0.4169	0.4333	**0.4421**	2.04%
	N@20	0.2495	0.2296	0.2493	0.2501	0.2513	0.2540	**0.266**	4.72%

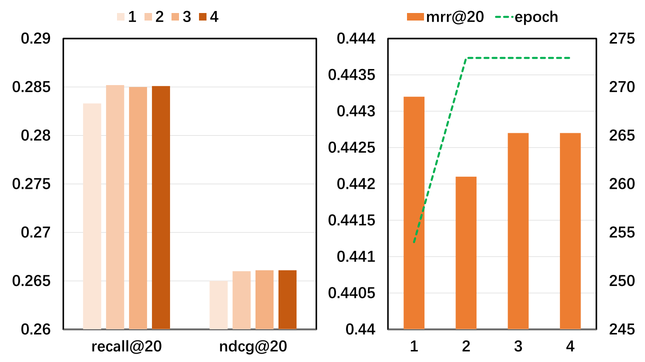

FIGURE 3. Performance comparison of setting the different number of order on Movielens-1M. 1, 2, 3, 4 denotes order=1,2,3,4. Experiments are shown when the number of Cube Unit is 1 and dropout=0.5.

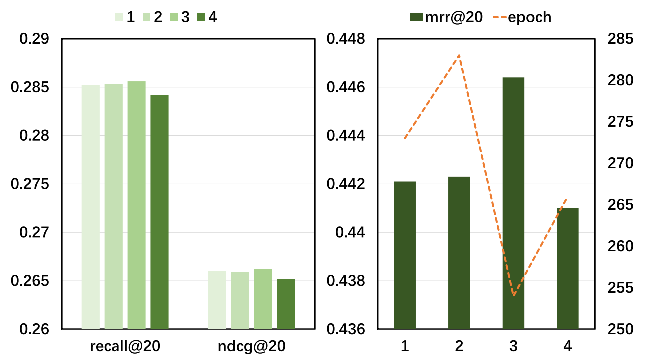

FIGURE 4. Performance comparison of setting the different number of Cube Unit on Movielens-1M. 1, 2, 3, 4 denotes Cube Unit's number=1,2,3,4. Experiments are shown when order=2 and dropout=0.5.

through the subsequent fully connected layers, and as order increases, the user and item embeddings accumulate their powers. In this set of ablation experiments, we set the range of order to [1,2,3,4] and use IcaGCN-order-1 to represent the result when order=1, and so on for other cases. In FIGURE 3, we can get the following conclusions:

- The increase of order can optimize the model's results on both recall and ndcg metrics to some extent, but at the same time it further increases the epoch required for training. simply, IcaGCN-order-2, IcaGCN-order-3 and IcaGCN-order-4 have better results than IcaGCN-order-1 on recall@20 and ndcg@20, IcaGCN-order-1 has the best performance in the training process because of less computation and shorter training time when order=1.
- Starting from IcaGCN-order-3, the gain of IcaGCN with increasing order is no longer significant and the required epoch is stabilized at 273. If the user embedding and item embedding are performed too many powers, it does not have much impact on the training time, but it can no longer bring gain to the model.

2) EFFECT OF CUBE UNIT

To better test the role of Cube Units and explore the importance of their depth in mining nodes in the global context, we repeated the Cube Units for a number of times in the range [1,2,3,4]. It should be noted that we use IcaGCN-Cube-1 to refer to the results where the number of Cube Units is 1, and so on for other cases. The experimental results under Movielens-1M dataset are shown in FIGURE 4. We have the following findings:

- We can clearly observe that the results of IcaGCN-Cube-1 and IcaGCN-Cube-2 are similar under the three metrics of recall@20, mrr@20 and ndcg@20, while the results of IcaGCN-Cube-3 are significantly better than those of IcaGCN-Cube-1 and IcaGCN-Cube-2, especially mrr@20.
- IcaGCN-Cube-3 not only has excellent results on the above three metrics, but also the number of epochs required for training is significantly less than the other three experiments, which indicates that when increasing the number of Cube Units, it has little effect on the single-round training time, and can even shorten the total training time to some extent.
- Both the above three metrics and the number of epochs required for training, IcaGCN-Cube-4 does not perform as well as IcaGCN-Cube-3. Therefore, we believe that increasing the number of Cube Units will not bring better

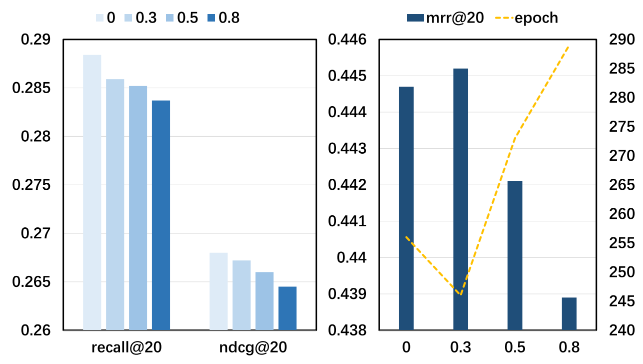

FIGURE 5. Performance comparison of different dropout on Movielens-1M. 0, 0.3, 0.5, 0.8 denotes dropout=[0,0.3,0.5,0.8]. Experiments are shown when order=2 and the number of Cube Unit is 1.

results, and the performance of IcaGCN-Cube-3 is the peak of the four sets of experiments.

3) IMPACT OF DROPOUT

We tested the results of IcaGCN with different dropout. Dropout takes values in the range of [0,0.3,0.5,0.8], and when dropout=0, it means that no nodes are discarded and all of them are included in the next calculation. FIGURE 5 shows our results on Movielens-1M dataset, and the following findings are obtained:

- For both recall@20 and ndcg@20 metrics, they gradually become smaller as the dropout increases, i.e., the number of nodes discarded increases. For mrr@20, it reaches its maximum value when dropout=0.3, and the result is similar for dropout=0. However, this result tends to decrease when dropout is greater than or equal to 0.5. In addition, when comparing the number of epochs needed for training, we find that the epoch increases gradually as dropout increases. the difference between the epochs at dropout=0 and dropout=0.3 is only 10 rounds, which is only 1 minute in the actual training. Therefore, we can argue that for data sets with sparse data and small data volume like Movielens-1M, a larger dropout is not suitable, and all nodes should be retained as much as possible to avoid time and metric losses.

4) DATA ENHANCEMENT CAPABILITY

In order to further demonstrate the data enhancement ability of Coactivate Unit in the IcaGCN model, we selected LightGCN as the basis, and then added three data enhancement methods of random negative sampling (RNS) [20], dynamic negative sampling (DNS) [31] and markov chain monte carlo negative sampling (MCNS) [29] respectively to conduct experiments on the two datasets. The experimental results are shown in TABLE 3, our model maintains better performance.

5) EFFICIENCY COMPARISON

To further demonstrate the speed advantage of the IcaGCN model, we compare the training time and epochs of

TABLE 3. Comparison of data enhancement capabilities.

Dataset	Amazon-Books			Yelp2018		
Metrics	R@20	M@20	N@20	R@20	M@20	N@20
LightGCN+RNS	0.1123	0.0778	0.0638	0.1134	0.0772	0.0628
LightGCN+DNS	0.1188	0.1005	0.0701	0.1155	0.0804	0.0630
LightGCN+MCNS	0.1164	0.0995	0.0659	0.1152	0.0803	0.0630
IcaGCN_order	**0.1189**	**0.1089**	**0.0704**	**0.1158**	**0.0807**	**0.0637**

TABLE 4. Efficiency comparison of different modules runs on Yelp2018 dataset.

Model	R@20	N@20	Time/Epoch	Epoch	Training Time
LightGCN	0.1134	0.0628	**82.26s**	274	6.22h
IcaGCN$_{order}$	0.1158	0.0637	83.22s	245	5.66h
IcaGCN$_{cube}$	0.1173	0.0642	86.96s	251	6.06h
IcaGCN	**0.1185**	**0.0656**	87.43s	**224**	**5.44h**

TABLE 5. Comparison between contrastive IcaGCN and LightGCN.

Dataset	Amazon-Books			Gowalla		
Metrics	R@20	M@20	N@20	R@20	M@20	N@20
LightGCN	0.1123	0.0778	0.0638	0.1898	0.1307	0.1106
SGL	0.1198	0.0853	0.0682	0.2066	0.1433	0.1236
LightGCN$_{sgl}$	0.1189	0.0876	0.0673	0.1950	0.1398	0.1196
LightGCN$_{bc-loss}$	0.1194	0.1057	0.0703	0.2044	0.1435	0.1235
IcaGCN	0.1194	0.1107	0.0717	0.2056	0.1442	0.1238
IcaGCN$_{sgl}$	0.1203	0.1114	0.0763	0.2061	0.1460	0.1244
IcaGCN$_{bc-loss}$	**0.1268**	**0.1189**	**0.0801**	**0.2093**	**0.1495**	**0.1268**

LightGCN and IcaGCN on the Yelp2018, as shown in TABLE 4. Among them, we also compared the recommendation effect and training time of the IcaGCN model with only 3-Layer Cube Units without order and only order=3 without Cube Unit. The IcaGCN model requires fewer epochs, reducing the total training time.

6) COMBINATION WITH CONTRASTIVE LEARNING

In this section, we'll explore how our approach combines with graphic-and-contrast learning on Amazon Books and Gowalla. We simply apply the comparative learning techniques of SGL [27] and BC-Loss [30] to IcaGCN. As shown in the TABLE 5, the model based on comparative learning has obtained significant benefits. Compared with contrastive learning-based collaborative filtering methods, our model still maintains better performance.

V. RELATED WORK

CNNs are often used to process images and speech data with a regular Euclidean structure, and they use a weighted summation to perform a convolutional operation that is spatially invariant in the face of distorted inputs but constant outputs. This is achieved by calculating the weighted sum of the central pixel points and the neighboring pixel points to form a feature map. In this case, the weight coefficients of the convolution kernel are used as weighting coefficients, the initial values are randomized, and then iterative optimization is performed by back-propagating the gradient descent according to the error function. Whether the convolution parameters can be optimized or not is very important for convolutional networks [10]. With the advent of the data era and the prosperity of social networks, we need to think about how to handle data with non-Euclidean space structure rationally, and GNNs and GCNs are born. GNNs play an active role in classification, regression and association prediction. When processing the next node, the information of its neighboring nodes is also considered; when looping once, the current node only aggregates the information of its own neighboring nodes, but in the next loop, it needs to aggregate the information of its neighboring nodes. In short, the input of GNN is the characteristics of each node itself and the relationship graph between nodes, and the output is the final representation of each node's characteristics after considering the graph structure. The main difference between GCNs and GNNs is the aggregation operation [32].

In 2018, when GAT [22] use an attention mechanism to assign different weights to each node, learn node features and structural features of nodes in the graph, and process local information while paying attention to the overall information. The core idea of GAT is to assign attention only to the first-order neighbors of the nodes. In 2019, NGCF [25] enhances embedding by explicitly modeling the high-order connectivity between user-item, and mining the higher-order connectivity relations to capture interactions to refine the multiple embedding propagation of embedding. Then in 2020, LightGCN' success shows that deforming from the standard GCN using a nonlinear activation function and feature transformation matrix is unuseful [7]. So it removes the nonlinear activation function and feature transformation matrix from NGCF, and only adds a set of weight coefficients to neighborhood aggregation of embeddings output from different GCN layers as the final embedding. Then, graph convolutional networks become a hot research direction, the research on heterogeneous graphs is also in full swing. For example, BiHGH [5] is a bidirectional graph convolution algorithm, which transfers knowledge sequentially by repeating up and down convolution, divides the four-partite graph into three subgraphs, and each subgraph only involves two adjacent entity types to design double similarity preservation regularization to prevent information loss in the process of hashing learning. PFCM [4] is an entity that can unify the three types of users, items and attributes and their relationships. By inheriting the content of user interaction items, PFCM learns user representation and implements metaphath-guided heterogeneous graph learning.

In recent years, negative sampling has been studied in the recommendation task of solving a class of problems. Specifically, most of the interaction between users and objects is implicit feedback. Bayesian Personalized Ranking (BPR) [20], which is one of the most commonly used methods, adopts uniform distribution for negative distribution of samples. Hard Negative Sampler adaptively picks the hardest negative by the current recommender and DNS [31] selects the negative scored highest by the recommender. MCNS [29] derived a theory to quantify the effect of negative sampling distribution and based on this, the positive distribution is approximated by self-contrast approximation.

Contrastive learning is a self-supervised learning method that learns the general characteristics of a data set by having the model learn which data points are similar or different. SGL [27] explores self-supervised learning on user-item graph to alleviate the long tail effect. Through hypergraph-enhanced cross-view contrast learning architecture, HCCF [28] can jointly capture local and global cooperative relationships, enhance the recognition ability of the CF paradigm based on GNN, and comprehensively capture the complex higher-order dependency relationships between users, effectively combining hypergraph structure coding with self-supervised learning. BC-Loss [30] incorporates the perceived margin of deviation into comparison losses, where the margin is quantitatively adjusted for the degree of deviation in each user-item interaction. BC-Loss can be used not only as a debias strategy, but also as a standard loss in recommendation models.

VI. CONCLUSION AND FUTURE WORK

In our work, we explore the high-order importance of user and item nodes and propose a model named IcaGCN which can efficiently model nodes' weights in a very flexible, plug-and-play way. Without losing the information of the data itself, more important nodes are given greater weight. Extensive experiments and ablation experiments on four real-world datasets have shown the high efficiency and speed of IcaGCN. To some extent, our work demonstrates the bright future of graph convolution networks in the field of recommendation systems and explores an efficient method to calculate the weight of all nodes, and to distinguish important and non-important nodes without losing the influence of original data as much as possible. In future work, we will explore more plug-and-play methods to overcome the adverse effects of some nodes, so as to improve the performance of the recommendation system.

REFERENCES

[1] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2017, *arXiv:1706.02263*.

[2] Y. Dai, F. Gieseke, S. Oehmcke, Y. Wu, and K. Barnard, "Attentional feature fusion," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 3560–3569.

[3] C. Gao, W. Lei, X. He, M. de Rijke, and T.-S. Chua, "Advances and challenges in conversational recommender systems: A survey," *AI Open*, vol. 2, pp. 100–126, 2021.

[4] W. Guan, F. Jiao, X. Song, H. Wen, C.-H. Yeh, and X. Chang, "Personalized fashion compatibility modeling via metapath-guided heterogeneous graph learning," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 482–491.

[5] W. Guan, X. Song, H. Zhang, M. Liu, C.-H. Yeh, and X. Chang, "Bi-directional heterogeneous graph hashing towards efficient outfit recommendation," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 268–276.

[6] M. H. Guo, T. X. Xu, J. J. Liu, Z. N. Liu, P. T. Jiang, T. J. Mu, S. H. Zhang, R. R. Martin, M. M. Cheng, and S. M. Hu, "Attention mechanisms in computer vision: A survey," *Comput. Vis. Media*, vol. 8, pp. 1–38, Mar. 2022.

[7] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 639–648.

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[9] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[10] T. Kattenborn, J. Leitloff, F. Schiefer, and S. Hinz, "Review on convolutional neural networks (CNN) in vegetation remote sensing," *ISPRS J. Photogramm. Remote Sens.*, vol. 173, pp. 24–49, Mar. 2021.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[12] Y. Koren, S. Rendle, and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*. Springer, 2022, pp. 91–142.

[13] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[14] B. Liu, R. Tang, Y. Chen, J. Yu, H. Guo, and Y. Zhang, "Feature generation by convolutional neural network for click-through rate prediction," in *Proc. World Wide Web Conf.*, May 2019, pp. 1119–1129.

[15] B. Liu, N. Xue, H. Guo, R. Tang, S. Zafeiriou, X. He, and Z. Li, "AutoGroup: Automatic feature grouping for modelling explicit high-order feature interactions in CTR prediction," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 199–208.

[16] B. Liu, C. Zhu, G. Li, W. Zhang, J. Lai, R. Tang, X. He, Z. Li, and Y. Yu, "AutoFIS: Automatic feature interaction selection in factorization models for click-through rate prediction," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 2636–2645.

[17] P. Massa and P. Avesani, "Trust metrics in recommender systems," in *Computing With Social Trust*. Cham, Switzerland: Springer, 2009, pp. 259–285.

[18] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: How accuracy metrics have hurt recommender systems," in *Proc. CHI Extended Abstracts Hum. Factors Comput. Syst.*, 2006, pp. 1097–1101.

[19] H. Pei, B. Wei, K. Chen-Chuan Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric graph convolutional networks," 2020, *arXiv:2002.05287*.

[20] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," 2012, *arXiv:1205.2618*.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.

[22] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *Stat*, vol. 1050, no. 20, pp. 10–48550, 2019.

[23] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3156–3164.

[24] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proc. ADKDD*, Aug. 2017, pp. 1–7.

[25] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 165–174.

[26] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, and T. Y. Liu, "A theoretical analysis of ndcg ranking measures," in *Proc. 26th Annu. Conf. Learn. Theory (COLT)*, vol. 8, 2013, p. 6.

[27] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 726–735.

[28] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang, "Hypergraph contrastive collaborative filtering," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2022, pp. 70–79.

[29] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou, and J. Tang, "Understanding negative sampling in graph representation learning," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 1666–1676.

[30] A. Zhang, W. Ma, X. Wang, and T.-S. Chua, "Incorporating bias-aware margins into contrastive loss for collaborative filtering," 2022, *arXiv:2210.11054*.

[31] W. Zhang, T. Chen, J. Wang, and Y. Yu, "Optimizing top-n collaborative filtering via dynamic negative item sampling," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2013, pp. 785–788.

[32] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.

[33] W. Xin Zhao, Y. Hou, X. Pan, C. Yang, Z. Zhang, Z. Lin, J. Zhang, S. Bian, J. Tang, W. Sun, Y. Chen, L. Xu, G. Zhang, Z. Tian, C. Tian, S. Mu, X. Fan, X. Chen, and J.-R. Wen, "RecBole 2.0: Towards a more up-to-date recommendation library," 2022, *arXiv:2206.07351*.

JINGXUE ZHANG was born in Hebei, China, in 1999. She received the bachelor's degree in software engineering from the Shanghai University of Technology, in 2021. From 2021 to 2023, she has been with the Data Mining Technology Laboratory, Changzhou University, where she is focusing on recommendation systems.

CHANGCHUN YANG was born in Jiangsu, China, in 1963. He received the bachelor's and master's degrees in data mining from the East China University of Science and Technology. He has published more than 50 academic papers, 19 textbooks, two national natural science foundation projects, and dozens of government and enterprise cooperation projects. His research interests include database systems, data mining, and big data analysis. He is a member of the Expert Group of the Teaching Guidance Sub-Committee of the Computer Science and Technology Specialty of the Ministry of Education, a Council Member of the China Computer Education Research Society, and a Senior Member of the China Computer Society.