A Graph Contrastive Learning with Feature Perturbation for Recommender Systems

Peihang Du, Jie Wu*, Chi Ma, Hui Hu, Yuenai Chen, and Jingyan Li

Abstract—**Recommender systems are an effective solution to address the issue of information overload and a thriving research field. This paper focuses on the efficient mining of user-item relationships and the aggregation of neighborhood information in bipartite graph-based recommender systems. Therefore, a Graph Contrastive Learning recommendation model with Feature Perturbation (GCL-FP) is proposed in this work. The proposed model utilizes a graph convolutional encoder with a residual structure, simplifying the feature transformation and the nonlinear activation in graph convolution. This approach helps alleviate the over-smoothing problem of node representation. Furthermore, a data augmentation method, incorporating graph convolution aggregation information into the random noise, is designed. By perturbing the graph node feature matrix in this way when generating the contrastive learning view, we ensure that the noise does not excessively alter the semantic information of node features. Finally, experiments on three benchmark datasets, namely Yelp2018, Douban-Book, and Alibaba-iFashion, were conducted. The experimental results show that our proposed model outperforms the baseline model Simple Graph Contrastive Learning (SimGCL) by improving the Recall evaluation metric by 2.2%, 5.4%, and 1.9%, respectively as well as the Normalized Discounted Cumulative Gain (NDCG) evaluation metric increases by 1.8%, 3.0%, and 1.8%, respectively.**

Index Terms—**recommender systems, graph contrastive learning, graph convolutional network, data augmentation**

I. Introduction

RECOMMENDER systems are information filtering systems that select items of interest based on the user behavior, aiming to maximize the user attraction and improve his engagement. Moreover, the key to achieve higher-level recommendations is learning high-quality representations of users and items [1-5]. As this topic is not new, early recommendation algorithms, such as matrix factorization, relied on collaborative filtering to mine the relationships between users and items [6-8]. These algorithms resulted in effective representations. However, for the past few years, Graph Convolutional Networks (GCN) have received increasing attention in the recommendation field due to their ability to apply high-order connectivity between nodes in order to learn high-quality representations [9, 10], leading to even better recommendation performance.

Although GCNs have achieved advanced performance, there are still many unresolved issues within this approach. The most two relevant limitations are listed here below:

(1) Over-smoothing phenomenon: Stacking multiple graph convolutional modules will make the node features, in the same connected area, much more similar, reducing therefore the discriminability of the node representations and causing the model to degrade. To solve this problem, researchers have introduced residual structures in the graph convolutional models, where the shallow and deep outputs are added as inputs for the next stage of learning to enhance the over-smoothing problem;

(2) Noise and sparsity in recommendation data: Recommendation data is constructed from the users' implicit feedback (e.g., clicks, likes, purchases) [6, 11], which can be noisy due to the user errors. To address this issue, researchers have introduced self-supervised learning, where the models are trained on a series of auxiliary tasks without the need for human-labeled labels [12, 13]. Furthermore, contrastive learning is a self-supervised learning algorithm that can alleviate difficulties in data labeling, noise interference, and data sparsity [14]. For instance, Wu *et al.* [15] proposed the Self-supervised Graph Learning (SGL) model, which uses contrastive learning as an auxiliary task to supplement the recommendation task. They achieved this by forming a contrasting view through random deletion of edges or nodes, effectively altering the graph structure in self-supervised tasks. However, the graph augmentation contrastive learning may not be generalized to various scenarios and may lose important structural information in the graph, such as changing the structure of a molecule. Therefore, Yu *et al.* [16] proposed a feature- augmentation Graph Contrastive Learning (GCL) method. This approach introduces random noise into the feature matrix to achieve augmentation at the graph representation level. While this approach preserves the graph structure, adding noise may alter the node features and lead to negative effects, such as the change of semantic information.

To address the aforementioned limitations, this paper introduces a novel recommendation model that integrates

Manuscript received May 16, 2023; revised September 5, 2023. This paper is supported by Foundation of Guangdong Educational committee under the Grant No. 2022ZDZX4052, 2021ZDJS082, No.2019KQNCX148.

Peihang Du is a postgraduate student of the school of Computer and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, Liaoning, China (e-mail: 1012134821@qq.com).

Jie Wu is an associate professor of the School of Computer and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, Liaoning, China (*corresponding author to provide e-mail:wujieaa@163.com).

Chi Ma is an associate professor of the School of Computer Science and Engineering, Huizhou University, Huizhou, 516007, Guangdong, China (e-mail: machi@hzu.edu.cn).

Hui Hu is an associate professor of the School of Computer Science and Engineering, Huizhou University, Huizhou, 516007, Guangdong, China (e-mail: Huhui@hzu.edu.cn).

Yuenai Chen is a lecturer of the School of Mathematics and Statistics, Huizhou University, Huizhou, 516007, Guangdong, China (e-mail: chenyuenai@163.com).

Jingyan Li is a lecturer of the School of Computer Science and Engineering, Huizhou University, Huizhou, 516007, Guangdong, China (e-mail: LJY@hzu.edu.cn).

GCNs with GCL methods. In the GCL task, a feature perturbation data augmentation method is designed to incorporate aggregated information, preserving the structure of the graph while minimizing the variations in the semantic information of node features.

Therefore, the main contributions of this work are as follows:

1) For the recommendation task, we design a GCN encoder with residual structures, which effectively alleviates the over-smoothing problem of the graph convolution and improves the performance of the model.

2) For the self-supervised auxiliary task, we design the GCL recommendation method on the basis of feature augmentation. This method applies random noise with an aggregated information to the feature matrix to form contrastive views, enabling the model to learn consistent embeddings while preserving the feature information without perturbance.

3) We conduct experimental studies on three benchmark datasets and evaluate the superiority of our proposed method.

II. Related Work

A. *Graph Convolutional Networks*

The GCN technique has been widely applied in several engineering domains, such as computer vision and natural language processing, due to its excellent performance in learning from graph data [17, 18]. Moreover, GCN leverages the graph structure to capture the complex relationships between objects. Its main focus is to perform convolutional operations on the graph to identify the node features, and update each node's feature representation by using the features of its neighbor nodes as filters [19]. The graph convolution operation is defined as follows:

$$H^{(l+1)} = \sigma(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^l W^l) \tag{1}$$

where, D is the degree matrix, A represents the adjacency matrix, σ indicates the non-linear activation function, W presents the weight matrix of the linear transformation, and H represents the feature representation at each layer.

In recommender systems, extracting collaborative signals from massive user and item interaction information is a challenging problem. The use of high-order connectivity to encode collaborative signals in the interaction graph structure is particularly suitable for the representation of recommendation relationships [20]. In addition, GCN was first applied in recommendation systems by Wang *et al.* [21] with their Neural Graph Collaborative Filtering (NGCF) method, which explicitly injects collaborative signals into node embeddings to improve the recommendation performance. Moreover, He *et al.* [22] proposed the recommendation algorithm LightGCN, which also applies the graph convolutional techniques to the recommendation systems and removes the feature transformation and the non-linear activation operations of the graph convolutional algorithm. Nevertheless, subsequent experiments have demonstrated that these operations have no substantial effect on the model, and the LightGCN model, not only improves recommendation performance, but also makes the whole model algorithm more lightweight.

B. *Graph Contrastive Learning*

In recent years, contrastive learning has achieved remarkable results in various engineering fields such as computer vision and natural language processing. Contrastive learning consists of a self-supervised learning method that mainly mines additional information from unsupervised data through a series of auxiliary tasks, to learn valuable representations for downstream tasks. This method has been applied in numerous scenarios, including social networks, protein interaction networks, molecular structures, and academic citation networks. As for GCL, which preserves the characteristics of graph data, it mainly consists of three modules: data augmentation, shared Graph Neural Network (GNN) encoder for learning graph representations, and contrastive loss. In the graph classification domain, You *et al.* [23] proposed the Graph Contrastive Learning (GraphCL) model. This model incorporates four graph contrastive learning data augmentation methods, including node dropping, edge perturbation, random walk, and attribute masking. The objective is to identify node embeddings by maximizing the similarity between two randomly perturbed versions of the same node's local subgraph representation. In addition, Zhu *et al.* [24] proposed the Graph Contrastive learning with Adaptive augmentation (GCA) model, which assigns larger removal probabilities to unimportant edges and larger mask probabilities to unimportant node feature dimensions. This will help in implementing contrastive learning and preserving important nodes and edges in the graph.

Furthermore, the recommended data is typically represented as a graph structure; however, it faces challenges, such as data sparsity and the long-tail effect caused by power-law distributions [25, 26]. As for the data augmentation techniques in GCL, it can effectively alleviate data sparsity, expand data volume, and serve as auxiliary tasks in recommendations to addressing such issues and enhancing node feature learning for improved model performance. In addition, the SGL model introduces GCL to the recommendation domain, proposing a new learning paradigm that employs graph data augmentation techniques, including node dropout and edge perturbation, to augment input data. This augmentation approach aims to reduce bias and enhance robustness to interaction noise. SimGCL also replaces the graph augmentation contrastive learning approach with a simple and efficient feature augmentation method. Referring to the experimental results, graph augmentation is not necessary; instead, using feature augmentation methods can produce more uniformly embedded representations of nodes, improving thereby the recommendation performance.

C. *Residual Networks*

In the computer vision domain, as model depth increases, the problem of model degradation often arises, where increasing network depth leads to a decrease at the level of performance. To address this issue, He *et al.* [27] proposed the Residual Network (ResNet), which introduces skip connections that add the input directly to the output. Even if the feature information is distorted or lost, the network can still recover the original feature information based on the residual connection. Introducing residual connections not only solves the problem of gradient vanishing but also enables networks, with hundreds of layers, to converge toward the optimal solution. Therefore, the equation that models the residual connection is represented as follows:

$$y = F(x) + x \quad (2)$$

where the input to the residual block is denoted by *x*, the transformed input is represented by $F(x)$, and *y* indicates the output of the residual connection.

In the field of natural language processing, Jiang *et al.* [28] proposed the Attention-based Relational Graph Convolutional Network (ARGCN) model, which combines GCNs with residual networks for Target-Oriented Opinion Words Extraction (TOOWE). The use of residual structures enhances the over-smoothing problem of convolutional graphs, thus improving the model performance. For instance, Dang *et al.* [29] proposed the Multi-Scale Residual Graph Convolution Network (MSR-GCN) model, which uses the Residual Graph Convolutional Network (RGCN) to predict the human motion by gradually abstracting the complex high-scale pose into low-scale poses and applying the residual structure between various input and output scales, resulting in a significant improvement in the model performance. To sum up, these studies suggest that incorporating residual structures into GCN can improve the overall performance of the system.

III. Proposed Model

A. *Problem Statement*

In recommendation systems, the relationship between users and items is usually defined as a bipartite graph $G=(U,V,\varepsilon)$, where $U=\{u_1,u_2,\ldots,u_n\}$ represents the set of users, $V=\{v_1,v_2,\ldots,v_m\}$ represents the set of items, and ε represents the set of edges. Let $e_u^{(0)} \in \mathbb{R}^d$ and $e_i^{(0)} \in \mathbb{R}^d$ be the initial embeddings of user *u* and item *i*, respectively. $R \in \mathbb{R}^{n\times m}$ is the matrix that records the interactions between users and items, where $R_{ui}=1$ indicates that user *u* has interacted with item *i*, and $R_{ui}=0$ indicates that there is no interaction between user *u* and item *i*. $E^{(0)}$ represents the initial feature matrix, $E^{(0)} \in \mathbb{R}^{(m+n)\times d}$, which serves as the input of the model. A graph convolutional based recommendation model can be represented as $e_u^{(k+1)} = AGG(e_u^{(k)}, \{e_i^{(k)} : i \in N_u\})$, which obtains the final node embeddings after *n* layers of graph convolutions. The model predicts the score $y=\{y_1,y_2,\ldots,y_n\}$ by computing the inner product of user nodes and item nodes after applying several graph convolution layers. Based on the scores, the top-k items are selected as the recommended items for the user.

B. *Overview*

In this study, we propose a model consisting of two modules, namely the recommendation task module and the contrastive learning task module. For these two modules, we design a lightweight RGCN with a shared GCN encoder to extract node features for score prediction. In the graph contrastive learning auxiliary task, we integrate random noise carrying aggregated information into the node features to create a contrastive view, resulting in the feature-level data augmentation. Note that in order to obtain high quality node representations, the contrastive learning task serves as an auxiliary task to complement the recommendation task. Finally, the overall architecture of the model is illustrated in Figure 1.

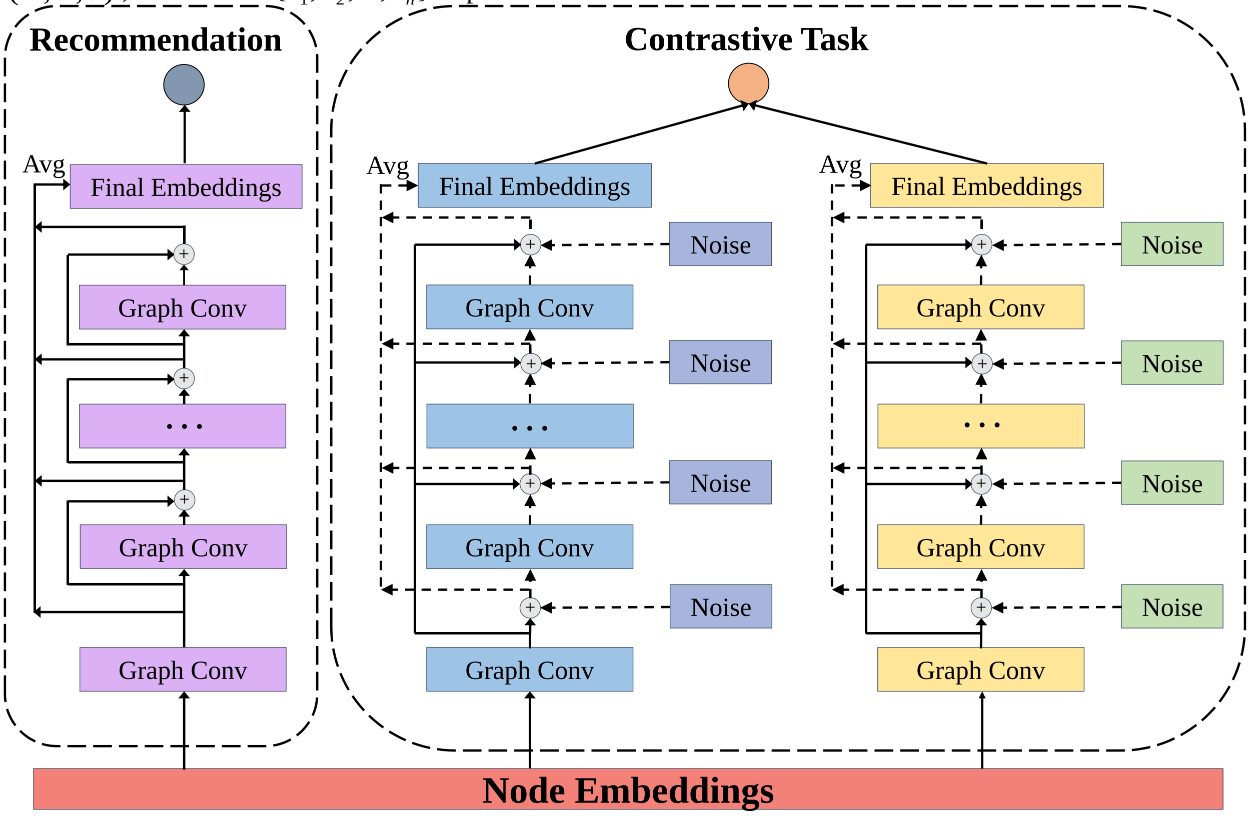

Fig. 1 The architecture of GCL-FP. The solid arrow refers the normal graph convolution information passing, and the dotted arrow refers to the graph convolution information passing after adding noise.

C. Recommendation Task

For the recommendation task, we designed a lightweight RGCN that serves as an encoder to extract node features. In the graph convolutional calculation, feature transformation and nonlinear activation were eliminated and residual structures were introduced, allowing the output of each layer to be added to the input of the subsequent one. After obtaining the node embeddings at each layer, a weighted average aggregator was used to generate the final representations for user and item nodes. Therefore, the graph convolution calculation is represented by the following equations:

$$e_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{N_u}\sqrt{N_i}} e_i^{(k)} + F(e_u^{(k)}) \tag{3}$$

$$e_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{N_i}\sqrt{N_u}} e_u^{(k)} + F(e_i^{(k)}) \tag{4}$$

Referring to Eq. (3), the left summation term serves as the inter-layer aggregation propagation term whereas the symmetric normalization term $1/\sqrt{N_u}\sqrt{N_i}$ addresses the problem of the embedding becoming excessively large due to the increase in the number of layers in the graph convolution operation. Also, the embedding representation of item *i*, obtained after *k*-layer GCN propagation, is denoted as $e_i^{(k)}$, where N_u represents the set of items that user *u* has interacted with, while N_i represents the set of users who have interacted with item *i*. In addition, $F(e_u^{(k)})$ represents the residual operation and $F(\cdot)$ is a standardization function, the output of the *k*-*th* GCN layer is passed through the Leaky Rectified Linear Unit (LeakyReLU) activation function and a standardization function. In a further stage, the result is added to the inter-layer aggregation information of the *k*-*th* GCN layer to generate the output of the (*k*+1)-*th* layer. After *n* layers of GCN aggregation, the final representation of each node is obtained.

Moreover, the model prediction is defined as the inner product of the final representations of the user and the item nodes. Therefore, the calculation formula is represented as follows:

$$y_{ui} = e_u^T e_i \tag{5}$$

The score, calculated using Eq. (5), is considered as the final predicted ranking score.

D. Graph Contrastive Learning Auxiliary Task

(1) Creating contrast views

For the graph contrastive learning auxiliary task, a feature perturbation data augmentation method is proposed. This method incorporates the aggregation information. In this work, we added random noise to the feature vectors to adjust the uniform representation of features within a certain range [16]. Additionally, we incorporated the neighborhood information, obtained through the graph convolutional layer aggregation, to ensure that the magnitude of the noise does not adversely affect the semantic information of the node features.

It should be noted that, in creating views for the graph contrastive learning task using a lightweight RGCN encoder, the strategy of consistently is adopted. This strategy integrates the residual connections into the embedding representations after the first layer of graph convolution calculation before adding noise. This is done, rather than integrating the residual connections into the embedding representation of the previous layer. Due to the modification of the original feature matrix caused by the introduction of noise, the performance is adversely affected.

Moreover, a graph representation-level data augmentation method is adopted. This method requires generating a random noise matrix after each layer of graph convolution computation and fuse it with the information obtained from the graph convolution aggregation of the current layer. Formally, a random noise matrix, with the same dimensions as the feature matrix after the *k*-*th* layer of graph convolution computation, is generated and added to the information obtained through the graph convolution aggregation using a certain fusion strategy. The calculation formula for this fusion process can be expressed as follows:

$$\Delta = \Delta^{(k)} + \eta E^{(k)} \tag{6}$$

where $E^{(k)}$ represents the output of the *k*-*th* GCN layer, with $E^{(k)} \in \mathbb{R}^{(n+m)\times d}$. Meanwhile, $\Delta^{(k)}$ denotes the randomly generated noise matrix for the *k*-*th* layer, which also satisfies $\Delta^{(k)} \in \mathbb{R}^{(n+m)\times d}$ and η indicates the hyperparameter that controls the amount of incorporating aggregated information $E^{(k)}$; moreover Δ is the final generated noise matrix. The purpose of incorporating the aggregated information $E^{(k)}$ into the random noise matrix is to ensure that the noise does not have an excessive impact on the semantic interpretation of the feature matrix. After creating the noise matrix, it is added to the feature matrix in the *k*-*th* layer to get the perturbed feature representation. Therefore, the calculation formula is modelled as follows:

$$E'^{(k)} = F(\Delta) + E^{(k)} \tag{7}$$

where $F(\cdot)$ is the normalization function, $E'^{(k)}$ is the feature matrix of the *k*-*th* layer GCN with added noise, aiming to achieve slight feature perturbation for each feature vector in $E^{(k)}$. By adding the random noise as a layer embedding after each graph convolution, a weighted average aggregator is used to get the comparison view E'. Similarly, another contrast view E'' is created, and it is considered as the input for the contrastive loss.

(2) Noise control

Adding excessive noise to the feature matrix can alter the information carried by the features, so the noise matrix needs to satisfy $\Delta = \bar{\Delta} \odot sign(E)$, $\bar{\Delta} \in \mathbb{R}^{(m+n)\times d} \sim U(0,1)$. In addition, taking the feature vector e_i of item *i* as an example, Δ_i' and Δ_i'' are two noise vectors that perturb e_i on two different views, it is required that $\|\Delta_i\|_2 = \mu$ and Δ_i are numerically equivalent to points on a hypersphere with a radius of μ, and that e_i, Δ_i', and Δ_i'' are in the same hyperoctant. The noise satisfying the above conditions does not lead to significant bias, equivalent to rotating the vectors by two small angles in the vector space, and each rotation corresponds to the deviation of the vector [16]. After generating noise and integrating graph convolution aggregation information to generate Δ'_i and Δ''_i, resulting in rotating the vectors towards e_i direction, getting e_i' and e_i''. This process controls the effect of noise on the features. Fig. 2 illustrates the difference between adding random noise (ne_i', ne_i'') and adding noise with aggregated information (e_i', e_i'').

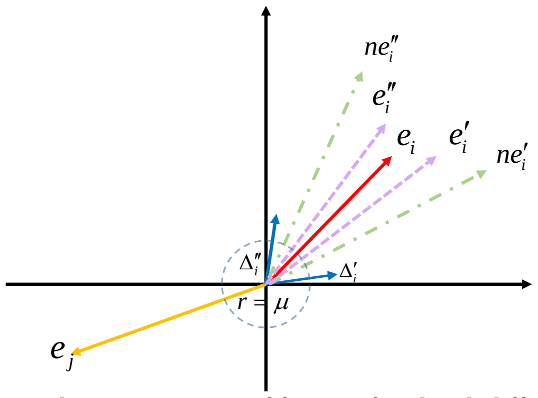

Fig. 2 The representation of features fused with different forms of noise in the vector space.

E. Module Training

The Graph Convolutional Learning with Feature Perturbation (GCL-FP) model adopts a multi-task training strategy, consisting of the recommendation task and the GCL auxiliary task. Each task has a separate loss function and is trained jointly with the other task.

Concerning the recommendation task, we used the Bayesian Personalized Ranking (BPR) loss function, which represents the user's historical behavior as a triplet. Given a user *u* and two items *i* and *j*, where *i* represents the item that the user has interacted with (*i.e.*, the positive sample) and *j* is an item that the user has not interacted with (*i.e.*, a negative sample), the goal is to make the predicted score for the positive sample *i* higher than that for the negative sample *j*. The BPR loss function is computed as follows:

$$L_{rec} = \sum_{(u,i,j)\in O} -\log\sigma(y_{ui} - y_{uj}) + \lambda\left\|\Theta\right\|^2 \tag{8}$$

For the Eq. (8), $O = \{(u,i,j) | (u,i) \in O^+, (u,j) \in O^-\}$, O^+ represents the observed interactions, while O^- represents the unobserved interactions. The σ indicates the sigmoid function and λ is the coefficient controlling L2 regularization. After obtaining the final predicted score, we optimize the model parameters using pairwise BPR loss, denoted as $\Theta = \{u, i | u \in U, i \in I\}$. In this model, the only trainable parameter is $E^{(0)}$, that is $\Theta = \{E^{(0)}\}$. Therefore, the formula can be derived as follows:

$$L_{rec} = \sum_{(u,i,j)\in O} -\log\sigma(y_{ui} - y_{uj}) + \lambda\left\|E^{(0)}\right\|^2 \tag{9}$$

In the graph contrastive learning auxiliary task, the Information Noise Contrastive Estimation (InfoNCE) contrastive loss function is used to maximize the consistency between positive samples and minimize the consistency between negative samples. In other words, the distance between the positive samples becomes closer and the distance between negative samples becomes farther in the vector space. The equation that models the user's self-supervised loss calculation is as follows:

$$L_{ssl}^{user} = \sum_{u\in U} -\log\frac{\exp(s(z_u', z_u'')/\tau)}{\sum_{v\in U}\exp(s(z_u', z_v'')/\tau)} \tag{10}$$

where *u* and *v* represent a user and an item in a sampled batch, z_u' and z_u'' are the data-augmented representations of user *u*, $s(\cdot)$ represents the cosine similarity, and τ is a hyperparameter defined as the temperature coefficient. Similarly, the self-supervised loss calculation formula for item L_{ssl}^{item} can be obtained. Adding the two losses together gives the objective function of the self-supervised task, denoted by $L_{ssl} = L_{ssl}^{user} + L_{ssl}^{item}$.

By adopting a multi-task learning strategy, the recommendation task loss and the self-supervised auxiliary task loss are jointly optimized for both tasks. Hence, the calculation formula is expressed as follows:

$$L_{loss} = L_{rec} + L_{ssl} + \lambda_2\left\|E^{(0)}\right\|^2 \tag{11}$$

For the Eq. (11), as a complete contrastive learning loss function.

IV. Experiments

A. Datasets

We conducted experiments on three public benchmark datasets to evaluate the performance of our model. These datasets are described below:

Yelp2018 [22]: Yelp is a well-known business review website in the United States, where users can rate and review businesses, and share shopping experiences with others.

Douban-Book [16]: This dataset is a collection of user-generated book reviews and ratings from the popular Chinese social networking site, Douban. The dataset regroups books in various languages, including Chinese and English, and is commonly used for natural language sentiment analysis as well as recommendation system research.

Alibaba-iFashion [15]: The third dataset is a large-scale fashion image dataset created by researchers at Alibaba Group. It supports research in the fields of fashion image analysis and recommendation retrieval. Despite its sparsity, this dataset is still large in scale.

TABLE I
The statistics of the datasets.

Dataset	#User	#Item	#Interaction
Yelp2018	31,668	38,048	1,561,406
Douban-Book	13,024	22,347	792,062
Alibaba-iFashion	300,000	81,614	1,607,813

The statistical data for the three datasets is shown in Table I

B. Baseline

We compare our proposed GCL-FP model to the following models:

1) **LightGCN** [22] proposes a lightweight graph convolution approach that stops feature transformation and non-linear activation, yielding to a great reduction in the number of parameters and improvement in the training efficiency.
2) **DNN+SSL** [30] proposes a large-scale self-supervised learning framework for recommendations. This approach employs a Deep Neural Net (DNN) as the item encoder and applies two types of enhancement methods, namely the feature masking and the feature dropout, to the existing item features.
3) **SGL** [15] introduces GCL into the recommendation domain by using three types of graph data augmentation to construct the contrastive views for recommendation.
4) **BUIR** [31] has a dual-branch architecture consisting of a target network and an online network. Moreover, it only

uses positive examples for self-supervised learning while ignoring long-tail items.

5) **MixGCF** [32] designs a single-hop mixing to combine local graph messages to generate informative negatives samples and improve recommendation generalization.
6) **NCL** [33] is a novel contrastive model that proposes a prototype contrastive objective to capture the relevance between the relationship of users/items and their context.
7) **SimGCL** [16] is a simple graph contrastive learning framework that doesn't use graph augmentation but adds random noise to node features.

C. Settings

The performance of our model was evaluated using two popular metrics in the recommendation field, namely Recall@20 and NDCG@20, to assess the performance of the proposed model. Multiple experiments were conducted on the GCL-FP model, To ensure a fair comparison of different models, the optimal hyperparameter settings were used based on the original papers that served as reference. Among all the baseline models, the embeddings were initialized using the Xavier initialization method with a size of 64 and L2 regularization parameter set to 10^{-4} to mitigate overfitting. Moreover, the batch size was set to 2048. To optimize all models, we used the Adam optimizer with a learning rate of 0.001 to achieve the best performance. Finally, concerning the SimGCL and SGL models, $\tau = 0.2$ was initiated as the temperature value.

D. Comparison with Baseline Methods

All methods were evaluated using two popular evaluation metrics in the recommendation field; moreover, the experimental results are presented in Table II. The SGL model employed the edge-dropout method, which achieved the best performance in its original paper. However, the obtained results showed that the performance of the GCL-FP model outperformed other baseline models, confirming the rationality and effectiveness of introducing the residual structure graph convolutional encoder as well as the feature perturbation graph contrastive learning method along with the aggregated information.

Based on the evaluation results proposed in Table II, we can observe the performance of other recommendation strategies. Firstly, MixGCF shows significant improvement in the evaluation of the metrics by enhancing the negative sampling strategy based on the LightGCN model, demonstrating therefore its superiority. Secondly, SGL performs better than LightGCN, reflecting that the use of GCL methods is efficient in the recommendation domain. Thirdly, NCL proposes a novel graph contrastive learning perspective that considers the neighbors of users (or items) from both the graph structure and the semantic space, achieving a comparable performance to SGL. Lastly, SimGCL uses feature augmentation to generate contrastive views, outperforming the SGL model on all three datasets while training faster, which confirms the theoretical proposition in the SimGCL paper that graph augmentation is unnecessary.

Our GCL-FP model did not add more parameters; thus, the training efficiency is comparable to SimGCL. By adding residual connections in the graph convolutional encoder, the network can identify the residual between the input and output, instead of trying to learn the complete representation of each layer from scratch. This would help to enhance the over-smoothing problem and stack more GCN layers to improve the model performance. Additionally, incorporating aggregate information in random noise can effectively achieve feature perturbation without changing the semantic information, while avoiding the time complexity of reconstructing the graph and without damaging its structure. Therefore, the data augmentation method has a positive impact on model learning.

In general, incorporating graph self-supervised learning as an auxiliary task in recommendation models performs better than using GCNs alone for recommendation. Thus, our proposed GCL-FP model achieved the best performance on both evaluation metrics across all three datasets.

E. Ablation Study

We conducted ablation experiments to compare our proposed GCL-FP model with the GCL-FPN that only adds random noise as well as the lightweight graph convolutional encoder GCL-FPGCN that does not include residual information. The experimental results, displayed in Table III, show that the performance of the GCL-FP model is superior to those of GCL-FPN and GCL-FPGCN. This indicates that the effectiveness of incorporating residual structures and aggregated noise to improve recommendation performance is validated.

TABLE II
MAIN EXPERIMENTAL RESULTS

Methods	Yelp2018		Douban-Book		Alibaba-iFashion	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN	0.0639	0.0525	0.1501	0.1282	0.1053	0.0505
DNN+SSL	0.0483	0.0382	0.1366	0.1148	0.0818	0.0375
SGL-ED	0.0675	0.0555	0.1732	0.1551	0.1093	0.0531
BUIR	0.0487	0.0404	0.1127	0.0938	0.0830	0.0384
MixGCF	0.0713	0.0589	0.1731	0.1552	0.1085	0.0520
NCL	0.0670	0.0562	0.1723	0.1545	0.1088	0.0528
SimGCL	0.0722	0.0598	0.1772	0.1578	0.1145	0.0548
GCL-FP	**0.0738**	**0.0609**	**0.1868**	**0.1643**	**0.1167**	**0.0558**

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT ARCHITECTURES

Methods	Yelp2018		Douban-Book		Alibaba-iFashion	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
GCL-FPN	0.0730	0.0602	0.1818	0.1596	0.1160	0.0554
GCL-FPGCN	0.0733	0.0603	0.1806	0.1595	0.1155	0.0553
GCL-FP	**0.0738**	**0.0609**	**0.1868**	**0.1643**	**0.1167**	**0.0558**

F. Comparison of Training Efficiency

In this section, we compared the required epochs and the total running time for model training on three datasets. The reported data is based on pytorch version 1.10 and it was collected using a GeForce RTX 3080Ti GPU. We set the number of layers in the graph convolution module to three.

As shown in Fig. 3, the LightGCN model requires significantly more training epochs compared to other models. Although SGL-ED requires considerably fewer training epochs than LightGCN, it still needs a higher number of epochs than other models when being applied on the Yelp2018 and Alibaba-iFashion datasets. Therefore, our GCL-FP model, in comparison to SimGCL, achieves a reduced number of epochs on the Yelp2018 and Douban-Book datasets, while maintaining a comparable number of training epochs when being applied on the Alibaba-iFashion dataset.

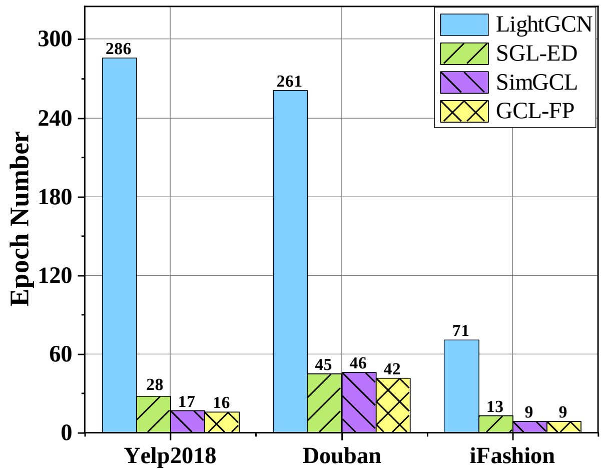

Fig. 3 The training epochs of compared methods.

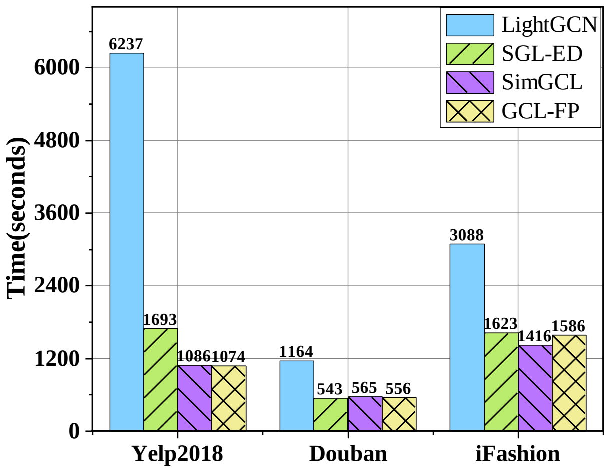

Fig. 4 The training time of compared methods.

Referring to Fig. 4, in terms of the total training time, LightGCN also requires a significantly longer time compared to other models, taking approximately twice their needed time or even more. As for SGL-ED, its total training duration on the Yelp2018 and Alibaba-iFashion datasets is longer than that of our model, while it is roughly on par with our model when dealing with the Douban-Book dataset. To sum up, our GCL-FP model, when being compared to SimGCL, exhibits similar training epochs on Alibaba-iFashion; however, its total training time is slightly higher than that of SimGCL. On the other hand, for the two other datasets, the total training time is almost the same between both models.

Based on these findings, it can be observed that employing graph contrastive learning for recommendation can accelerate the model's convergence and reduce the training time. In addition, it is evident that the use of feature augmentation not only enhances the recommendation performance of the model but also results in higher training efficiency and faster convergence speed. Compared to SimGCL, which also uses the feature augmentation, our proposed model exhibits superior performance without adding any extra burden to the training process.

G. Hyperparameter Studies

In this paper, a lightweight graph convolutional network with residual connectivity is proposed to help improve the over-smoothing problem. However, too many GCN layers can still lead to low performance or significantly slower convergence. Therefore, hyperparameter experiments were conducted by varying the number of GCN layers from 1 to 7 to observe the effect on the model performance. As shown in Fig. 5, the experimental results indicate that the Yelp2018 dataset achieved the best performance when the GCN layers were set to five, while the Douban-Book dataset reached its peak performance with four GCN layers. Finally, the optimal performance for the Alibaba-iFashion dataset corresponded to three GCN layers, , which may be due to its sparsity.

Moreover, η as a hyperparameter controlling the amount of aggregated information fused with noise, we found that the size of the aggregated information incorporated into the noise can affect the model's learning ability for different datasets. To be more exact, if too much aggregated information is added, the generated contrastive views may have too little diversity, leading to a poor performance. On the other hand, if the value is too small, it may alter the semantic information of the features. Therefore, we adjust the parameter value η to observe changes in performance. As shown in Fig. 6, we found that a unit value for η achieves the optimal performance for Yelp2018 dataset, while a value of 0.8 for Douban-Book dataset and 0.4 for Alibaba-iFashion dataset reached the best performances.

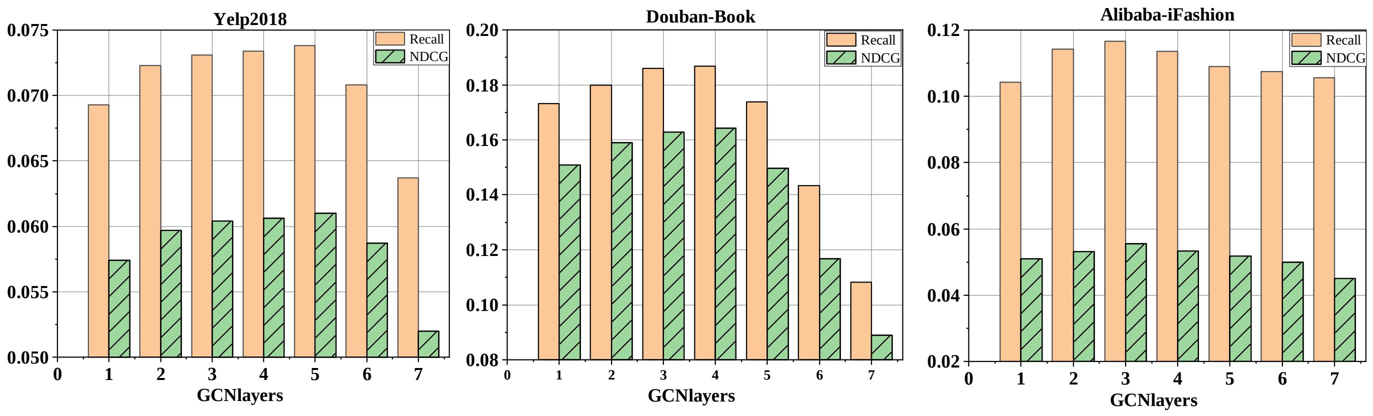

Fig. 5 Influence of the magnitude GCN layers on model performance.

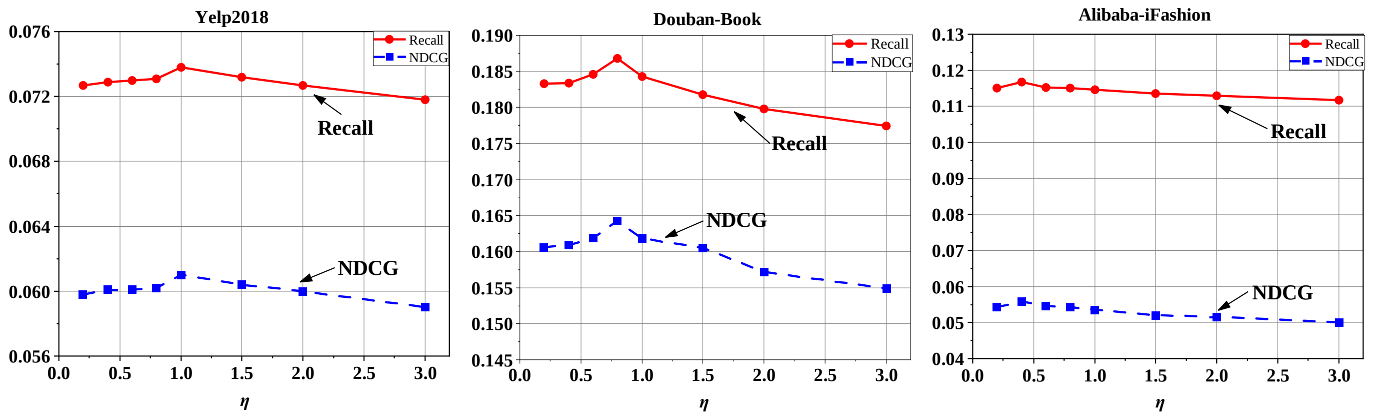

Fig. 6 Influence of the magnitude η of CL.

V. Conclusion

In this work, we propose a recommendation method based on graph convolutional networks and graph contrastive learning. For the shared GCN encoder, we design a lightweight graph convolutional neural network with residual structure that serves as an encoder to extract node features. This encoder accelerates the model convergence speed and improves its performance. Moreover, we incorporate random noise as the aggregation information into the feature matrix to create contrastive views. The random noise introduces slight perturbations to the feature matrix, not only enabling the node features to learn uniform representations but also preserving the semantic information of the features; thus, it may assist the recommendation task in learning high-quality representations. Finally, the experimental results show that the GCL-FP model outperforms other representative models and effectively improves the recommendation performance.

References

[1] P. Gu, Y. Han, W. Gao, G. Xu, and J. Wu, “Enhancing session-based social recommendation through item graph embedding and contextual friendship modeling,” Neurocomputing, vol.419, pp. 190-202, 2021.

[2] A. S. Tewari, and A. G. Barman, “Sequencing of items in personalized recommendations using multiple recommendation techniques,” Expert Systems with Applications, vol.97, pp. 70-82, 2018.

[3] X. Wu, Y. Li, J. Wang, Q. Qian, and Y. Guo, “UBAR: User Behavior-Aware Recommendation with knowledge graph,” Knowledge-Based Systems, vol.254, pp. 109661, 2022.

[4] W. Pan, and K. Yang, “Enhanced Multi-Head Self-Attention Graph Neural Networks for Session-based Recommendation,” Engineering Letters, vol.30, no.1, pp. 37-44, 2022.

[5] Y.-E. Hou, W. Gu, K. Yang, and L. Dang, “Deep reinforcement learning recommendation system based on gru and attention mechanism,” Engineering Letters, vol.31, no.2, pp. 695-701, 2023.

[6] M. F. Aljunid, and M. D. Huchaiah, “Multi-model deep learning approach for collaborative filtering recommendation system,” Caai Transactions on Intelligence Technology, vol.5, no.4, pp. 268-275, 2020.

[7] L. J. M. P. I. E. Xueting, “Personalized Recommendation Algorithm of Tourist Attractions Based on Transfer Learning,” Mathematical Problems in Engineering, vol.2022, 2022.

[8] K. Miura, M. Takeuchi, and Y. Okada, “A recommender system based on an improved simultaneous selection method of query items and neighbors,” IAENG International Journal of Computer Science, vol.43, no.4, pp. 406-410, 2016.

[9] L. Huang, Y. Ma, Y. Liu, B. Danny Du, S. Wang, and D. J. a. T. O. I. S. Li, “Position-enhanced and time-aware graph convolutional network for sequential recommendations,” ACM Transactions on Information Systems, vol.41, no.1, pp. 1-32, 2023.

[10] K. Liu, F. Xue, X. N. He, D. Guo, and R. C. Hong, “Joint Multi-Grained Popularity-Aware Graph Convolution Collaborative Filtering for Recommendation,” IEEE Transactions on Computational Social Systems, vol.10, no.1, pp. 72-83, 2023.

[11] J. F. Wang, Z. Y. Fu, M. X. Niu, P. B. Zhang, and Q. L. Zhang, “Multi-feedback Pairwise Ranking via A dversarial Training for Recommender,” Chinese Journal of Electronics, vol.29, no.4, pp. 615-622, 2020.

[12] S. Chen, J. H. Xue, J. L. Chang, J. Z. Zhang, J. F. Yang, and Q. Tian, “SSL plus plus : Improving Self-Supervised Learning by Mitigating the Proxy Task-Specificity Problem,” IEEE Transactions on Image Processing, vol.31, pp. 1134-1148, 2022.

[13] Y. H. Tao, M. Gao, J. L. Yu, Z. W. Wang, Q. Y. Xiong, and X. Wang, “Predictive and Contrastive: Dual-Auxiliary Learning for Recommendation,” IEEE Transactions on Computational Social Systems pp. 1-12, 2022.

[14] X. Chen, Y. H. Pan, and B. Luo, “Research on power-law distribution of long-tail data and its application to tourism

recommendation," Industrial Management & Data Systems, vol.121, no.6, pp. 1268-1286, 2021.

[15] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, et al., "Self-supervised graph learning for Recommendation," Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, pp. 726-735, 2021

[16] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for Recommendation," Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1294-1303, 2022

[17] L. Q. Chen, P. Shi, G. H. Li, and T. Qi, "Traffic flow prediction using multi-view graph convolution and masked attention mechanism," Computer Communications, vol.194, pp. 446-457, 2022.

[18] J. H. Wang, Y. Guo, Z. H. Wang, Q. F. Tang, and X. X. Wen, "Advancing Graph Convolution Network with Revised Laplacian Matrix," Chinese Journal of Electronics, vol.29, no.6, pp. 1134-1140, 2020.

[19] L. Pasa, N. Navarin, and A. Sperduti, "Polynomial-based graph convolutional neural networks for graph classification," Machine Learning, vol.111, no.4, pp. 1205-1237, 2022.

[20] Z. Q. Pan, and H. H. Chen, "Efficient Graph Collaborative Filtering via Contrastive Learning," Sensors, vol.21, no.14, 2021.

[21] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, pp. 165-174, 2019

[22] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for Recommendation," Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, pp. 639-648, 2020

[23] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," Advances in neural information processing systems, vol.33, pp. 5812-5823, 2020.

[24] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," Proceedings of the Web Conference 2021, pp. 2069-2080, 2021

[25] J. W. Niu, L. Wang, X. T. Liu, and S. Yu, "FUIR: Fusing user and item information to deal with data sparsity by using side information in recommendation systems," Journal of Network and Computer Applications, vol.70, pp. 41-50, 2016.

[26] S. W. Wu, F. Sun, W. T. Zhang, X. Xie, and B. Cui, "Graph Neural Networks in Recommender Systems: A Survey," Acm Computing Surveys, vol.55, no.5, 2023.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016

[28] J. Jiang, A. Wang, and A. Aizawa, "Attention-based relational graph convolutional network for target-oriented opinion words extraction," Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 1986-1997, 2021

[29] L. Dang, Y. Nie, C. Long, Q. Zhang, and G. Li, "Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction," Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 11467-11476, 2021

[30] T. Yao, X. Yi, D. Z. Cheng, F. Yu, T. Chen, A. Menon, et al., "Self-supervised learning for large-scale item recommendations," Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4321-4330, 2021

[31] D. Lee, S. Kang, H. Ju, C. Park, and H. Yu, "Bootstrapping user and item representations for one-class collaborative filtering," Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 317-326, 2021

[32] T. Huang, Y. Dong, M. Ding, Z. Yang, W. Feng, X. Wang, et al., "Mixgcf: An improved training method for graph neural network-based recommender systems," Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 665-674, 2021

[33] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," Proceedings of the ACM Web Conference 2022, pp. 2320-2329, 2022