NLGCL: Naturally Existing Neighbor Layers Graph Contrastive Learning for Recommendation

Jinfeng Xu
The University of Hong Kong
Hong Kong, Hong Kong
jinfeng@connect.hku.hk

Zheyu Chen
The Hong Kong Polytechnic University
Hong Kong, Hong Kong
zheyu.chen@connect.polyu.hk

Shuo Yang
The Univerisity of Hong Kong
Hong Kong, Hong Kong
shuoyang.ee@gmail.com

Jinze Li
The University of Hong Kong
Hong Kong, Hong Kong
lijinze-hku@connect.hku.hk

Hewei Wang
Carnegie Mellon University
Pittsburgh, USA
heweiw@andrew.cmu.edu

Wei Wang
Shenzhen MSU-BIT University
Shenzhen, China
ehomewang@ieee.org

Xiping Hu
Beijing Institute of Technology
Beijing, China
huxp@bit.edu.cn

Edith Ngai*
The University of Hong Kong
Hong Kong, Hong Kong
chngai@eee.hku.hk

Abstract

Graph Neural Networks (GNNs) are widely used in collaborative filtering to capture high-order user-item relationships. To address the data sparsity problem in recommendation systems, Graph Contrastive Learning (GCL) has emerged as a promising paradigm that maximizes mutual information between contrastive views. However, existing GCL methods rely on augmentation techniques that introduce semantically irrelevant noise and incur significant computational and storage costs, limiting effectiveness and efficiency.

To overcome these challenges, we propose NLGCL, a novel contrastive learning framework that leverages naturally contrastive views between neighbor layers within GNNs. By treating each node and its neighbors in the next layer as positive pairs, and other nodes as negatives, NLGCL avoids augmentation-based noise while preserving semantic relevance. This paradigm eliminates costly view construction and storage, making it computationally efficient and practical for real-world scenarios. Extensive experiments on four public datasets demonstrate that NLGCL outperforms state-of-the-art baselines in effectiveness and efficiency.

CCS Concepts

• **Information systems** → **Recommender systems;**.

Keywords

Recommender System, Contrastive Learning

*Corresponding authors

RecSys '25, Prague, Czech Republic

ACM ISBN 979-8-4007-1364-4/25/09
https://doi.org/10.1145/3705328.3748059

ACM Reference Format:
Jinfeng Xu, Zheyu Chen, Shuo Yang, Jinze Li, Hewei Wang, Wei Wang, Xiping Hu, and Edith Ngai. 2025. NLGCL: Naturally Existing Neighbor Layers Graph Contrastive Learning for Recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25), September 22–26, 2025, Prague, Czech Republic.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3705328.3748059

1 Introduction

The proliferation of the internet has led to an explosion of information, making recommender systems indispensable in modern society, including domains such as social media [9, 15, 49], e-commerce [8, 33, 47], and short video platforms [11, 43]. Traditional recommender systems typically model user preferences based on historical user-item interactions [17, 46]. With advances in graph representation learning [20, 39], Graph Neural Networks (GNNs) have been introduced to extract collaborative signals from high-order neighbors, enhancing representation learning in recommender systems [17, 23, 37, 56]. Despite their popularity and effectiveness, learning high-quality user and item representations remains challenging due to the data sparsity problem and label dependency. To mitigate data sparsity, self-supervised learning (SSL) offers a promising avenue by generating supervised signals from unlabeled data [24, 25]. Among SSL paradigms, contrastive learning (CL) [18] has shown potential by maximizing mutual information between positive pairs in contrastive views, thereby leveraging self-supervised signals to enhance learning. Recently, graph contrastive learning (GCL) methods [40, 50, 51] have garnered significant attention in the recommender system field. Typically, GCL methods construct contrastive views using data augmentation techniques such as node deletion, edge perturbation, feature masking, and noise addition [1, 28]. For example, SGL [40] applies stochastic perturbations to nodes and edges, while SimGCL [51] and LightGCL [2] manipulate graph structures via noise addition and singular value decomposition, respectively. Other methods like NCL [22] use clustering algorithms, and DCCF

[29] and BIGCF [54] focus on disentangled representations through SSL.

However, existing GCL methods have significant limitations regarding both **effectiveness** and **efficiency**, which are discussed and validated in Section 3 by comprehensive investigation. **Effectiveness**: While augmentation techniques enable models to learn robust and invariant features, they can introduce semantically irrelevant noise [40, 51] that distorts critical structural and feature information. Random alterations may not preserve the underlying semantics of the graph, potentially hindering model performance [22, 29]. **Efficiency**: Constructing and storing augmented contrastive views increases computational and storage overhead, leading to higher time complexity and reduced scalability [2, 54]. In practical scenarios where quick responses and resource efficiency are crucial, these additional costs limit the applicability of existing methods.

To this end, we propose a simple yet effective paradigm called Neighbor Layers Graph Contrastive Learning (NLGCL). Unlike existing GCL methods relying on data augmentation, NLGCL leverages naturally existing contrastive views within GNNs. Specifically, NLGCL treats each node and its neighbors in the next layer as positive pairs, and each node and other nodes as negative pairs. In the recommendation field, graphs are constructed from historical user-item interactions. Users and items with similar interaction histories share similar semantic information after neighbor aggregation in GNNs. By utilizing the naturally existing contrastive views between neighbor layers, NLGCL avoids introducing irrelevant noise and eliminates the need for additional augmented views, reducing computational costs. We conduct extensive experiments to validate the superiority of our NLGCL over various state-of-the-art baselines in terms of effectiveness and efficiency.

2 Preliminary

2.1 Definition

In this section, we provide some essential notations and definitions. The historical user-item interactions can be naturally represented as a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V}$ includes user nodes $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and item nodes $\mathcal{I} = \{i_1, i_2, ..., i_{|\mathcal{I}|}\}$. The edge set $\mathcal{E}$ consists of undirected edges between user nodes and item nodes. The user-item interaction matrix is denoted as $\mathcal{R} \in \{0, 1\}^{|\mathcal{U}|\times|\mathcal{I}|}$. Specifically, each entry $\mathcal{R}_{u,i}$ indicates whether the user u is connected to item i, with a value of 1 representing a connection and 0 otherwise. The entire user-item interaction matrix can be divided into an observed user-item interaction matrix $\mathcal{R}^+ \in \{\mathcal{R}_{u,i} | u \in \mathcal{U}, i \in \mathcal{I}, \mathcal{R}_{u,i} = 1\}$ and an unobserved user-item interaction matrix $\mathcal{R}^- \in \{\mathcal{R}_{u,i} | u \in \mathcal{U}, i \in \mathcal{I}, \mathcal{R}_{u,i} = 0\}$. It is clear that the number of undirected edges $|\mathcal{E}|$ equals the number of observed user-item interactions $|\mathcal{R}^+|$ in the training data. We random initialize $\mathbf{E}_u \in \mathbb{R}^{d\times|\mathcal{U}|}$ and $\mathbf{E}_i \in \mathbb{R}^{d\times|\mathcal{I}|}$ to represent user and item embeddings, respectively. The graph structure of $\mathcal{G}$ can be denoted as the adjacency matrix $\mathcal{A} \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times(|\mathcal{U}|+|\mathcal{I}|)}$, formally:

$$\mathcal{A} = \begin{bmatrix} 0^{|\mathcal{U}|\times|\mathcal{U}|} & \mathcal{R} \\ \mathcal{R}^T & 0^{|\mathcal{I}|\times|\mathcal{I}|} \end{bmatrix}. \tag{1}$$

The symmetrically normalized matrix is $\tilde{\mathcal{A}} = \mathcal{D}^{-\frac{1}{2}}\mathcal{A}\mathcal{D}^{-\frac{1}{2}}$, where $\mathcal{D}$ represents diagonal degree matrix.

2.2 GNNs for Recommendation

GNNs update node representation through aggregating messages from their neighbors. The core of the graph-based CF paradigm consists of two steps: **S1: message propagation** and **S2: node representation aggregation**. Thus, message propagation for the user/item node can be formulated as: $\mathbf{e}_u^{(l)} = \text{Aggr}^{(l)}(\{\mathbf{e}_i^{(l-1)} : i \in \mathcal{N}_u\})$ and $\mathbf{e}_i^{(l)} = \text{Aggr}^{(l)}(\{\mathbf{e}_u^{(l-1)} : u \in \mathcal{N}_i\})$, where $\mathcal{N}_u$ and $\mathcal{N}_i$ denote the neighborhood set of nodes u and i, respectively, and l denotes the l-th layer of GNNs. Then, the final user/item embeddings can be formulated as: $\bar{\mathbf{E}}_u = \text{Readout}([\mathbf{E}_u^{(0)}, \mathbf{E}_u^{(1)}, ..., \mathbf{E}_u^{(L)}])$ and $\bar{\mathbf{E}}_i = \text{Readout}([\mathbf{E}_i^{(0)}, \mathbf{E}_i^{(1)}, ..., \mathbf{E}_i^{(L)}])$, where the Readout$(\cdot)$ function can be any differentiable function, and L is the layer number of GNN. In the recommendation scenario, LightGCN [17] is currently the most popular GNN backbone, which effectively captures high-order information through neighborhood aggregation. For efficient training and to ensure that diverse neighborhood information is integrated, final embeddings are aggregated from all layers: $\bar{\mathbf{E}}_u = \frac{1}{L+1}(\mathbf{E}_u^{(0)} + \tilde{\mathcal{A}}^1\mathbf{E}_u^{(0)} + ... + \tilde{\mathcal{A}}^L\mathbf{E}_u^{(0)})$ and $\bar{\mathbf{E}}_i = \frac{1}{L+1}(\mathbf{E}_i^{(0)} + \tilde{\mathcal{A}}^1\mathbf{E}_i^{(0)} + ... + \tilde{\mathcal{A}}^L\mathbf{E}_i^{(0)})$, where $\bar{\mathbf{E}}_u$ and $\bar{\mathbf{E}}_i$ denote final embeddings for user and item, respectively. Subsequently, we derive scores for all unobserved user-item pairs using the inner product of the final embeddings of the user and item, denoted as $y_{u,i} = \bar{\mathbf{e}}_u^\top \bar{\mathbf{e}}_i$, where $\bar{\mathbf{e}}_u$ and $\bar{\mathbf{e}}_i$ represent the final representations of user u and item i, respectively. The items with the top-N highest score are recommended to the user.

To provide effective item recommendations from user-item interactions, a typical training objective is the pair-wise loss function. We take the most widely adopted BPR [31] loss as an example:

$$\mathcal{L}_{bpr} = \sum_{(u,p,n)\in O} -\ln\sigma(y_{u,p} - y_{u,n}), \tag{2}$$

where $O = \{(u, p, n) \mid (u, p) \in \mathcal{R}^+, (u, n) \in \mathcal{R}^-\}$ denotes the pair-wise training data, $\sigma(\cdot)$ denotes sigmoid function. Essentially, BPR aims to widen the predicted preference margin between the positive item p and negative item n for user u.

2.3 CL for Recommendation

Recent studies [2, 40, 50, 51] have demonstrated that CL, through the generation of self-supervised signals, effectively mitigates the challenge of the data sparsity problem in recommender systems. CL-based methods construct contrastive views through various data augmentation strategies and optimize the mutual information between contrastive views, thereby obtaining self-supervised signals. By maximizing the consistency among positive samples and minimizing the consistency among negative samples, most of the existing CL methods mainly adopt InfoNCE [26] for optimization:

$$\mathcal{L}_{cl} = -\sum_{u\in\mathcal{U}} \log\frac{\exp(\frac{\hat{\mathbf{e}}_u^\top\tilde{\mathbf{e}}_u}{\tau})}{\sum_{v\in\mathcal{U}}\exp(\frac{\hat{\mathbf{e}}_u^\top\tilde{\mathbf{e}}_v}{\tau})} - \sum_{i\in\mathcal{I}} \log\frac{\exp(\frac{\hat{\mathbf{e}}_i^\top\tilde{\mathbf{e}}_i}{\tau})}{\sum_{j\in\mathcal{I}}\exp(\frac{\hat{\mathbf{e}}_i^\top\tilde{\mathbf{e}}_j}{\tau})}, \tag{3}$$

where $\hat{\mathbf{e}}_u/\hat{\mathbf{e}}_i$ and $\tilde{\mathbf{e}}_u/\tilde{\mathbf{e}}_i$ represent representation of item/user in different contrastive views. τ denotes the temperature hyper-parameter. However, existing CL-based approaches inevitably require the construction of multiple views, which imposes significant computational overheads. Therefore, we raise a question: **Is it necessary to**

construct contrastive views? To answer this question, we provide an investigation in Section 3.

3 Investigation

In this section, we analyze existing CL methods and their computational costs, identify key limitations, and emphasize the naturally existing contrastive views in GNNs. These views eliminate the need for costly view construction while preserving essential information and avoiding irrelevant noise.

3.1 Computational Cost Analysis

Existing studies on CL emphasize the critical role of data augmentation in creating diverse contrastive views necessary for effective implementation [2, 22, 40, 50, 51, 53, 54]. In recommendation, there are two types: (1) graph augmentation and (2) representation augmentation. Graph augmentation typically involves generating augmented graphs by randomly discarding edges or nodes from the original graph: $\mathcal{G}' = \text{Dropout}(\mathcal{G}(\mathcal{V}, \mathcal{E}), p)$, where $\mathcal{G}$ denotes the original graph and $\mathcal{G}'$ is the augmented graph, and p represents the keep rate. Representation augmented introduces noise into the graph convolution process: $\mathbf{e}' = \mathbf{e} + \Delta$, where $\mathbf{e}$ denotes embedding, Δ denotes the added noise. While these contrastive views are effective, data augmentation inevitably compromises the quality of embeddings: (1) Graph augmentation methods, which discard edges when generating contrastive views, result in relatively poor embedding features and density distributions. (2) Representation augmentation methods introduce noise into graph convolutions. Although the impact of some noises on distributions is less affected than that of graph augmentation, it still leads to less smooth density distributions. Furthermore, data augmentation is computationally intensive: (1) Graph augmentation methods require generating multiple augmented graphs, which is highly time-consuming. (2) Representation augmentation methods necessitate repeated graph convolution or modeling operations to obtain contrastive views, further increasing computational time. In Table 4, we empirically analyze the time consumption of advanced CL-based models, supporting our observations.

3.2 Contrastive Learning Analysis

Constructing contrastive views is the primary source of computational cost in CL-based models. Therefore, we analyze the role of view construction in contrastive learning. Existing works [50, 51] suggest that CL enables the recommendation to learn more uniformly distributed user and item representations, thereby mitigating the prevalent popularity bias in CF. Specifically, the added noise in the contrastive views propagates as part of the gradient. The uniformity is then regularized to a higher level by sampling the noise from a uniform distribution. However, we state that existing CL methods commonly face two serious problems.

P1: As shown in Figure 1 (a), existing CL methods on two contrastive views can be seen as pulling each node closer to its representation in the other view and arbitrarily pushing it farther away from all other nodes in the other view, which can lead to nodes that have similar semantics being irrationally pushed farther away.

P2: Existing CL methods construct contrastive views by adding random noise, which inevitably discards crucial information and introduces irrelevant noise during view construction.

To address these challenges, we aim to compensate for the representation of similar nodes (**P1**) and construct semantically relevant contrastive views (**P2**). Interestingly, we find that multiple groups of such contrastive views naturally exist within GCNs. These views not only share explicit semantic relevance but also implicitly compensate for the representation of similar nodes during training.

3.3 Naturally Contrastive Views within GCN

We point out that heterogeneous nodes in neighbor layers constitute contrastive views (e.g., $(\mathbf{E}_u^{(0)}; \mathbf{E}_i^{(1)})$ and $(\mathbf{E}_i^{(0)}; \mathbf{E}_u^{(1)})$). To substantiate this, we first formulate the message-passing in GCNs:

$$\mathbf{e}_u^{(l)} = \sum_{\tilde{i} \in \mathcal{N}_u} \frac{\mathbf{e}_{\tilde{i}}^{(l-1)}}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_{\tilde{i}}|}}, \quad \mathbf{e}_i^{(l)} = \sum_{\tilde{u} \in \mathcal{N}_i} \frac{\mathbf{e}_{\tilde{u}}^{(l-1)}}{\sqrt{|\mathcal{N}_i|\,|\mathcal{N}_{\tilde{u}}|}}, \tag{4}$$

where $\mathcal{N}_u$ and $\mathcal{N}_i$ denote neighbor sets for user u and item i, respectively. $\mathcal{N}_{\tilde{i}}$ and $\mathcal{N}_{\tilde{u}}$ denote neighbor sets for item $\tilde{i} \in \mathcal{N}_u$ and user $\tilde{u} \in \mathcal{N}_i$, respectively. Note that all users in $\mathcal{N}_{\tilde{i}}$ have the same interacted item $\tilde{i}$ with user u and all items in $\mathcal{N}_{\tilde{u}}$ have the same interacted user $\tilde{u}$ with item i. Then we rewrite the message-passing in GCNs:

$$\mathbf{e}_{\tilde{i}}^{(l)} = \sum_{\tilde{u} \in \mathcal{N}_{\tilde{i}}} \frac{\mathbf{e}_{\tilde{u}}^{(l-1)}}{\sqrt{|\mathcal{N}_{\tilde{i}}|\,|\mathcal{N}_{\tilde{u}}|}}, \quad \mathbf{e}_{\tilde{u}}^{(l)} = \sum_{\bar{i} \in \mathcal{N}_{\tilde{u}}} \frac{\mathbf{e}_{\bar{i}}^{(l-1)}}{\sqrt{|\mathcal{N}_{\tilde{u}}|\,|\mathcal{N}_{\bar{i}}|}}. \tag{5}$$

Since item $\tilde{i} \in \mathcal{N}_u$ and user $\tilde{u} \in \mathcal{N}_i$, user u and i are involved in $\mathcal{N}_{\tilde{i}}$ and $\mathcal{N}_{\tilde{u}}$, respectively. By associating Eq (4) and Eq (5), we know that representation $\mathbf{e}_{\tilde{i}}^{(l)}$ of item $\tilde{i}$ in (l)-th layer can be regarded as a weighted aggregation of the representation $\mathbf{e}_u^{(l-1)}$ of user u in $(l-1)$-th layer and the representation $\mathbf{e}_{\bar{u}}^{(l-1)}$ of other users $\bar{u} \in \mathcal{N}_{\tilde{i}}$ in $(l-1)$-th layer who interact with item $\tilde{i}$. In contrast, representation $\mathbf{e}_{\tilde{u}}^{(l)}$ of user $\tilde{u}$ in (l)-th layer can be regarded as an equal-weighted aggregation of the representation $\mathbf{e}_i^{(l-1)}$ of item i in $(l-1)$-th layer and the representation $\mathbf{e}_{\bar{i}}^{(l-1)}$ of other items $\bar{i} \in \mathcal{N}_{\tilde{u}}$ in $(l-1)$-th layer who interact with user $\tilde{u}$. Users with similar preferences generally share a part of semantics, aligning with the GCN's goal to enhance representations by aggregating neighbor nodes. Thus the representations $\mathbf{e}_u^{(l-1)}$ and $\mathbf{e}_i^{(l-1)}$ of user u and item i in $(l-1)$-th layer form positive pairs with the representations $\mathbf{e}_{\tilde{i}}^{(l)}$ and $\mathbf{e}_{\tilde{u}}^{(l)}$ of their neighbor nodes items $\tilde{i} \in \mathcal{N}_u$ and users $\tilde{u} \in \mathcal{N}_i$ in (l)-th layer, respectively. Thus, we find that heterogeneous nodes in neighbor layers naturally constitute contrastive views. For each $(l-1)$-th layer user u, the items $\tilde{i} \in \mathcal{N}_u$ in (l)-th layer that interact with it are treated as positive samples, and the other items $\hat{i} \in (\mathcal{I} \setminus \mathcal{N}_u)$ in (l)-th layer that without interact with it are treated as negative samples. Similar tendencies are presented for items. For a clearer understanding, we present the naturally existing contrastive views between neighbor layers in Figure 1 (b).

We argue that leveraging natural contrastive views removes computational overhead and addresses the limitations of existing CL methods in Section 3.2. For **P1**, in the $(l-1)$-th layer view, positive

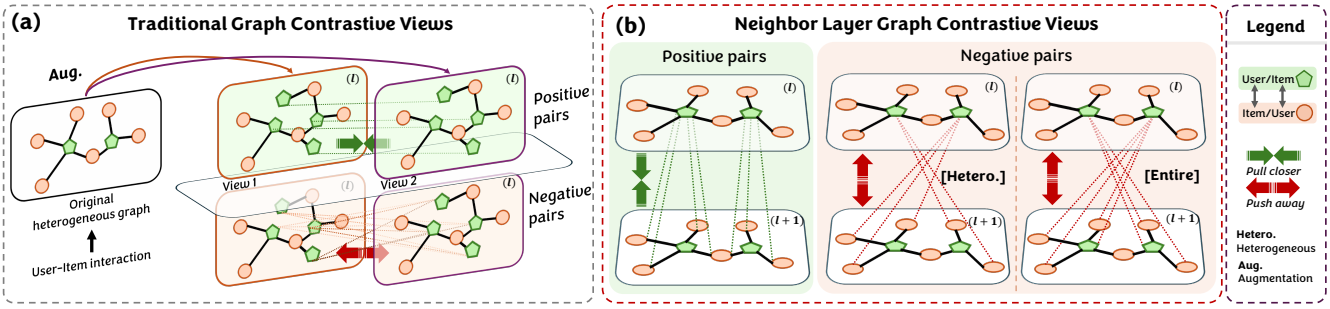

Figure 1: Overview of contrastive learning. Left: traditional contrastive learning paradigm; Right: our NLGCL.

samples in the neighbor layer l are aggregated from the representations of both the node itself and semantically similar nodes, compensating for the negative impact of arbitrarily treating semantically similar nodes as negative samples in traditional CL. For **P2**, while nodes with similar preferences share semantics, individual personalization leads to differences that GCNs exploit to enhance representations through neighbor aggregation. These semantic differences can be viewed as beneficial noise in constructing contrastive views, enhancing rather than hindering representation learning.

4 Methodology

In this section, we introduce a novel CL-based recommendation paradigm called Neighbor Layers Graph Contrastive Learning (NLGCL). Specifically, we first introduce the naturally existing contrastive views in neighbor layers. Then, we detailed our tailored loss function. Finally, a deeper analysis of our NLGCL is further proposed.

4.1 Contrastive Views

We use LightGCN as our graph convolution backbone, obtaining node embeddings $\mathbf{e}_u^{(0)}/\mathbf{e}_i^{(0)}, \mathbf{e}_u^{(1)}/\mathbf{e}_i^{(1)}, ..., \mathbf{e}_u^{(L)}/\mathbf{e}_i^{(L)}$, at each layer, as in Eq (4). We introduce two scopes of contrastive views: (a) **heterogeneous** and (b) **entire**. In the heterogeneous scope, based on the analysis in Section 3.3, all item embeddings in layer $(l-1)$ and all user embeddings in layer l form one pair of naturally contrastive views; conversely, all user embeddings in layer $(l-1)$ and all item embeddings in layer l form another pair. In the entire scope, we consider all embeddings in layer $(l-1)$ and all embeddings in layer l as naturally contrastive views. Next, we define the positive and negative pairs within these contrastive views.

4.1.1 Positive Pairs. Two scopes have the same positive pairs. Given a user u, the neighbor set for u is $\mathcal{N}_u$. For user u, the (l)-th layer embeddings of u's neighbors construct the positive pairs with the $(l-1)$-th layer embedding $\mathbf{e}_u^{(l-1)}$ of user u. Specifically, $\mathbf{e}_u^{(l-1)}$ constructs positive pairs with $\mathbf{e}_{\tilde{i}}^{(l)}$, where $\tilde{i} \in \mathcal{N}_u$. The reason for this because $\mathbf{e}_{\tilde{i}}^{(l)}$ are weighted aggregated by $\mathbf{e}_{\bar{u}}^{(l-1)}$, where $\bar{u} \in \mathcal{N}_{\tilde{i}}$ (as Eq (5)). Similar positive pairs are defined for item i.

4.1.2 Negative Pairs. In the heterogeneous scope, the (l)-th layer embeddings of all items, excluding items $\tilde{i}$, construct negative pairs with the $(l-1)$-th layer embedding $\mathbf{e}_u^{(l-1)}$ of user u, where $\tilde{i} \in \mathcal{N}_u$. Similar negative pairs are defined for item i. In the entire scope, the (l)-th layer embeddings of all users additionally construct negative pairs with the $(l-1)$-th layer embedding $\mathbf{e}_u^{(l-1)}$ of user u, while still maintains all negative pairs in the heterogeneous scope. Similar negative pairs are defined for item i. We categorize the positive and negative pairs in Table 1.

Table 1: Positive and Negative pairs in Contrastive Views.

Node	Scope	Positive Pairs	Negative Pairs
u: $\mathbf{e}_u^{(l-1)}$	Heterogeneous	$\mathbf{e}_{\tilde{i}}^{(l)} \vert \tilde{i} \in \mathcal{N}_u$	$\mathbf{e}_{\hat{i}}^{(l)} \vert \hat{i} \notin \mathcal{N}_u$
u: $\mathbf{e}_u^{(l-1)}$	Entire	$\mathbf{e}_{\tilde{i}}^{(l)} \vert \tilde{i} \in \mathcal{N}_u$	$\mathbf{e}_{\hat{i}}^{(l)} \cup \mathbf{e}_{\hat{u}}^{(l)} \vert \hat{i} \notin \mathcal{N}_u, u \in \mathcal{U}$
i: $\mathbf{e}_i^{(l-1)}$	Heterogeneous	$\mathbf{e}_{\tilde{u}}^{(l)} \vert \tilde{u} \in \mathcal{N}_i$	$\mathbf{e}_{\hat{u}}^{(l)} \vert \hat{u} \notin \mathcal{N}_i$
i: $\mathbf{e}_i^{(l-1)}$	Entire	$\mathbf{e}_{\tilde{u}}^{(l)} \vert \tilde{u} \in \mathcal{N}_i$	$\mathbf{e}_{\hat{u}}^{(l)} \cup \mathbf{e}_{\hat{i}}^{(l)} \vert \hat{u} \notin \mathcal{N}_i, i \in \mathcal{I}$

Analysis: Unlike traditional contrastive learning, our approach associates each node with multiple positive samples, where noise among positives arises from semantically similar nodes. This enhances the alignment of positive samples while reducing the impact of random noise. Additionally, it addresses the limitation of traditional CL, which arbitrarily treats semantically similar nodes as negatives. Both scopes use the same positive sample selection, but the entire scope provides a broader range of negatives. While a larger negative sample scope can improve positive feature learning, it may hinder semantic alignment between users and items and increase computational costs. We empirically validate the effectiveness and efficiency of both scopes in Sections 5.3 and 5.4.

4.2 Contrastive Learning Loss

As discussed in Section 3.2, traditional CL considers a node and its counterpart in another view as positive pairs, while treating the node and all other nodes in different views as negative pairs. However, having a small number of positive pairs and arbitrarily defined negative pairs can irrationally push nodes with similar semantics farther away. In contrast, our tailored CL method assigns each node a set of positive pairs, enhancing alignment and reducing the impact of random noise. In an L-layer GNN, up to L groups of contrastive views can be constructed, but this incurs additional computational overhead. Let G denote the number of contrastive view groups, where $G \leq L$. Selecting these G groups is crucial, and

we theoretically prove in Theorem 1 (Section 7) that the first G groups are optimal.

Theorem 1. *In GCNs, contrastive views that naturally exist between layer (l) and layer (l+1) are more effective for contrastive learning when (l) is smaller.*

Based on Theorem 1, selecting the first G layers and their neighbor layers as contrastive views achieves optimal results. We introduce our tailored CL loss for two scopes of contrastive views, respectively.

4.2.1 Heterogeneous Scope. For the heterogeneous scope, our neighbor layer CL loss can be formulated as:

$$\mathcal{L}_{nl_u} = -\frac{1}{G|\mathcal{U}|}\sum_{g=0}^{G-1}\sum_{u\in\mathcal{U}}\frac{1}{|\mathcal{N}_u|}\log\frac{\prod_{i^+\in\mathcal{N}_u}\exp(\mathbf{e}_u^{(g)\top}\mathbf{e}_{i^+}^{(g+1)}/\tau)}{\sum_{\hat{i}\in\mathcal{I}}\exp(\mathbf{e}_u^{(g)\top}\mathbf{e}_{\hat{i}}^{(g+1)}/\tau)}, \quad (6)$$

$$\mathcal{L}_{nl_i} = -\frac{1}{G|\mathcal{I}|}\sum_{g=0}^{G-1}\sum_{i\in\mathcal{I}}\frac{1}{|\mathcal{N}_i|}\log\frac{\prod_{u^+\in\mathcal{N}_i}\exp(\mathbf{e}_i^{(g)\top}\mathbf{e}_{u^+}^{(g+1)}/\tau)}{\sum_{\hat{u}\in\mathcal{U}}\exp(\mathbf{e}_i^{(g)\top}\mathbf{e}_{\hat{u}}^{(g+1)}/\tau)}, \quad (7)$$

where positive samples i^+/u^+ refer to all neighbors of each u/i, while negative samples $\hat{i}/\hat{u}$ are drawn from the entire set of $\mathcal{I}$ or $\mathcal{U}$. We compute the average over the first G layers for all users/items.

4.2.2 Entire Scope. For the entire scope, our neighbor layer CL loss can be formulated as:

$$\mathcal{L}_{nl_u} = -\frac{1}{G|\mathcal{U}|}\sum_{g=0}^{G-1}\sum_{u\in\mathcal{U}}\frac{1}{|\mathcal{N}_u|}\log\frac{\prod_{i^+\in\mathcal{N}_u}\exp(\mathbf{e}_u^{(g)\top}\mathbf{e}_{i^+}^{(g+1)}/\tau)}{\sum_{\hat{x}\in\mathcal{V}}\exp(\mathbf{e}_u^{(g)\top}\mathbf{e}_{\hat{x}}^{(g+1)}/\tau)}, \quad (8)$$

$$\mathcal{L}_{nl_i} = -\frac{1}{G|\mathcal{I}|}\sum_{g=0}^{G-1}\sum_{i\in\mathcal{I}}\frac{1}{|\mathcal{N}_i|}\log\frac{\prod_{u^+\in\mathcal{N}_i}\exp(\mathbf{e}_i^{(g)\top}\mathbf{e}_{u^+}^{(g+1)}/\tau)}{\sum_{\hat{x}\in\mathcal{V}}\exp(\mathbf{e}_i^{(g)\top}\mathbf{e}_{\hat{x}}^{(g+1)}/\tau)}, \quad (9)$$

where positive samples i^+/u^+ refer to all neighbors of each u/i, while negative samples $\hat{x}$ are drawn from the entire set $\mathcal{V} = \mathcal{I} \cup \mathcal{U}$. We compute the average over the first G layers for all users/items.

For any scope, we get the final loss $\mathcal{L}_{nl}$ by summing both user-side loss $\mathcal{L}_{nl_u}$ and item-side loss $\mathcal{L}_{nl_i}$, formally: $\mathcal{L}_{nl} = \mathcal{L}_{nl_u} + \mathcal{L}_{nl_i}$.

Efficiency: Although our NLGCL introduces a computational complexity proportional to G, the cost of constructing contrastive views in traditional CL is significantly higher than the cost of computing the CL loss itself. By leveraging the naturally existing contrastive views within GCNs, our NLGCL eliminates the substantial overhead of view construction. Detailed analysis of time efficiency is provided in Section 6.

4.3 Model Optimization

To extract node representations in NLGCL, we adopt a multi-task training strategy that jointly optimizes the traditional BPR loss (Eq (2)) and our neighbor layers graph contrastive learning loss:

$$\mathcal{L} = \mathcal{L}_{bpr} + \lambda_1\mathcal{L}_{nl} + \lambda_2\|\Theta\|^2, \quad (10)$$

where λ_1 and λ_2 are balancing hyper-parameters of CL and L_2 regularization term, respectively. Θ denotes model parameters.

Table 2: Statistics of four datasets.

Dataset	# Users	# Items	# Interaction	Sparsity
Yelp	45,477	30,708	1,777,765	99.873%
Pinterest	55,188	9,912	1,445,622	99.736%
QB-Video	30,324	25,731	1,581,136	99.797%
Alibaba	300,001	81,615	1,607,813	99.993%

5 Experiments

In this section, we briefly describe our experimental settings and then conduct extensive experiments on four public datasets to evaluate our proposed NLGCL by answering the following research questions: **RQ1:** How does our proposed NLGCL perform in comparison to various state-of-the-art recommender systems? **RQ2:** How do the different scopes of our NLGCL impact its performance? **RQ3:** How efficient is our NLGCL compared with various state-of-the-art recommender systems? **RQ4:** Can our NLGCL learn high-quality representations with uniform distribution? **RQ5:** How do different hyper-parameters influence?

5.1 Experimental Settings

5.1.1 Datasets. To validate the effectiveness of NLGCL, we conduct experiments on four public datasets, namely **Yelp** [22], **Pinterest** [13], **QB-Video** [52], and **Alibaba** [6]. These datasets originate from diverse domains, exhibiting variances in both scale and sparsity. To ensure the quality of the data, we employ the 15-core setting for the **Yelp** and **Alibaba** datasets, which ensures a minimum of 15 interactions between users and items. For the **Pinterest** and **QB-Video** datasets, users and items with less than 5 interactions are filtered out. Table 2 summarizes statistical information for all datasets. We use the user-based split approach to divide the data for experimentation. Specifically, we divide training, validation, and test sets into a ratio of 8:1:1, respectively. Moreover, we employ pair-wise sampling, where a negative interaction is randomly chosen for each interaction in the training set to construct the training sample.

5.1.2 Metrics. To evaluate the top-K recommendation task performance fairly, we adopt two widely-used metrics: Recall and Normalized Discounted Cumulative Gain (NDCG). The values of K are set to 10, 20, 50. Following previous work [22], we adopt the all-rank protocol which ranks all candidate items that users have not interacted with during evaluation.

5.1.3 Implementation Details. To ensure a fair comparison, we implement our NLGCL[2] and all baselines using the RecBole and RecBole-GNN [55], unified frameworks for traditional recommendation models and graph-based traditional recommendation models, respectively. We use the Adam optimizer [19] and Xavier initialization [14] with default parameters, and conduct extensive hyper-parameter tuning for all baselines. For our NLGCL, we fix the batch size to 4096, set λ_2 for L_2 regularization to 10^{-4}, and use an embedding size of 64. Early stopping is applied with an epoch limit of 20, monitored by NDCG@10. For our NLGCL, we tune hyper-parameters $\lambda_1 \in \{10^{-6}, 10^{-5}, 10^{-4}\}$, temperature $\tau \in$

[2]Code is available at: https://github.com/Jinfeng-Xu/NLGCL.

Table 3: Performance comparison of baselines and our NLGCL in terms of Recall@K(R@K) and NDCG@K(N@K). The superscript * indicates the improvement is statistically significant where the p-value is less than 0.01.

	Model	MF-BPR	NGCF	LightGCN	IMP-GCN	LayerGCN	SGL	NCL	SimGCL	LightGCL	DCCF	BIGCF	NLGCL	Improv.
	Metric	[30]	[37]	[17]	[23]	[56]	[40]	[22]	[51]	[2]	[29]	[54]	Our	
Yelp	R@10	0.0643	0.0630	0.0730	0.0751	0.0771	0.0833	0.0902	0.0908	0.0766	0.0818	0.0880	**0.0952***	4.85%
	R@20	0.1043	0.1026	0.1163	0.1182	0.1208	0.1288	0.1325	0.1331	0.1188	0.1268	0.1311	**0.1414***	6.24%
	R@50	0.1862	0.1864	0.2016	0.2017	0.2041	0.2100	0.2127	0.2130	0.2008	0.2107	0.2119	**0.2327***	9.25%
	N@10	0.0458	0.0446	0.0520	0.0539	0.0560	0.0601	0.0673	0.0682	0.0551	0.0596	0.0630	**0.0713***	4.55%
	N@20	0.0580	0.0567	0.0652	0.0662	0.0684	0.0739	0.0811	0.0823	0.0681	0.0741	0.0779	**0.0859***	4.37%
	N@50	0.0793	0.0784	0.0875	0.0885	0.0901	0.0964	0.1033	0.1039	0.0899	0.0974	0.1005	**0.1094***	5.29%
Pinterest	R@10	0.0855	0.0870	0.1000	0.0985	0.1004	0.1080	0.1033	0.1051	0.0881	0.1040	0.1040	**0.1148***	6.30%
	R@20	0.1409	0.1428	0.1621	0.1603	0.1620	0.1704	0.1609	0.1576	0.1322	0.1613	0.1619	**0.1793***	5.22%
	R@50	0.2599	0.2633	0.2862	0.2845	0.2880	0.2963	0.2887	0.2442	0.2383	0.2871	0.2894	**0.3089***	4.25%
	N@10	0.0537	0.0545	0.0635	0.0624	0.0635	0.0701	0.0666	0.0705	0.0534	0.0661	0.0680	**0.0760***	7.80%
	N@20	0.0708	0.0721	0.0830	0.0814	0.0826	0.0897	0.0833	0.0871	0.0673	0.0828	0.0864	**0.0948***	5.69%
	N@50	0.1001	0.1018	0.1136	0.1113	0.1121	0.1209	0.1150	0.1086	0.0981	0.1157	0.1155	**0.1267***	4.80%
QB-Video	R@10	0.1120	0.1261	0.1275	0.1278	0.1291	0.1341	0.1372	0.1337	0.1330	0.1350	0.1341	**0.1427***	4.01%
	R@20	0.1741	0.1943	0.1969	0.1966	0.1990	0.2030	0.2089	0.2030	0.2018	0.2044	0.2056	**0.2167***	3.73%
	R@50	0.2969	0.3237	0.3251	0.3259	0.3271	0.3342	0.3414	0.3356	0.3258	0.3281	0.3423	**0.3501***	2.28%
	N@10	0.0782	0.0888	0.0900	0.0899	0.0904	0.0932	0.0964	0.0945	0.0921	0.0951	0.0928	**0.0996***	3.32%
	N@20	0.0977	0.1099	0.1109	0.1105	0.1123	0.1141	0.1189	0.1151	0.1130	0.1150	0.1144	**0.1219***	2.52%
	N@50	0.1303	0.1438	0.1448	0.1447	0.1461	0.1482	0.1530	0.1501	0.1459	0.1508	0.1503	**0.1573***	2.81%
Alibaba	R@10	0.0303	0.0382	0.0457	0.0400	0.0448	0.0461	0.0477	0.0474	0.0459	0.0490	0.0502	**0.0531***	5.78%
	R@20	0.0467	0.0615	0.0692	0.0635	0.0680	0.0692	0.0713	0.0691	0.0716	0.0729	0.0744	**0.0773***	3.90%
	R@50	0.0799	0.1081	0.1144	0.1110	0.1138	0.1141	0.1165	0.1092	0.1204	0.1199	0.1205	**0.1238***	2.74%
	N@10	0.0161	0.0198	0.0246	0.0221	0.0238	0.0248	0.0259	0.0262	0.0239	0.0257	0.0266	**0.0299***	12.41%
	N@20	0.0203	0.0257	0.0299	0.0271	0.0285	0.0307	0.0319	0.0317	0.0305	0.0311	0.0322	**0.0361***	12.11%
	N@50	0.0269	0.0349	0.0396	0.0369	0.0393	0.0396	0.0409	0.0397	0.0410	0.0404	0.0415	**0.0450***	8.43%

Table 4: Efficiency comparison of different methods across four datasets, including average training time per epoch, number of epochs to converge, total training time (T/E: Time/Epoch, #E: #Epoch, TT: Total Time, N@10: NDCG@10; s: second, m: minute, h: hour). We highlight the optimal and suboptimal models for the TT metric in bold and underline, respectively.

Model	Yelp				Pinterest				QB-Video				Alibaba			
	T/E↓	#E↓	TT↓	N@10↑	T/E↓	#E↓	TT↓	N@10↑	T/E↓	#E↓	TT↓	N@10↑	T/E↓	#E↓	TT↓	N@10↑
LightGCN	22.42s	332	2h9m	0.0520	8.40s	194	27m	0.0635	25.32s	283	1h59m	0.0900	21.55s	371	2h13m	0.0246
SGL	104.19s	55	1h36m	0.0601	42.71s	43	31m	0.0701	124.44s	61	2h7m	0.0932	95.37s	83	2h12m	0.0248
NCL	94.42s	102	2h41m	0.0673	35.10s	83	49m	0.0666	108.98s	90	2h43m	0.0964	85.79s	88	2h6m	0.0477
SimGCL	98.33s	155	4h14m	0.0682	37.31s	121	1h15m	0.0705	110.92s	167	5h9m	0.0945	87.31s	241	5h51m	0.0262
LightGCL	100.51s	64	1h47m	0.0551	38.87s	51	33m	0.0534	119.18s	57	1h53m	0.0921	92.24s	69	1h46m	0.0459
DCCF	221.31s	38	2h20m	0.0596	76.39s	32	41m	0.0661	228.21s	38	2h25m	0.0951	180.08s	49	2h27m	0.0257
BIGCF	208.19s	41	2h22m	0.0630	70.11s	35	41m	0.0680	206.60s	38	2h11m	0.0928	167.12s	42	1h57m	0.0266
NLGCL-H	46.93s	63	**49m**	0.0713	19.55s	40	**13m**	0.0760	53.19s	55	**49m**	0.0996	44.73s	74	**55m**	0.0299
NLGCL-E	61.32s	82	1h28m	0.0689	28.81s	51	24m	0.0729	72.99s	62	1h15m	0.0969	59.67s	79	1h19m	0.0279

$\{0.1, 0.2, 0.3, 0.4\}$, the number of GCN layers $L \in \{1, 2, 3, 4\}$, and the number of contrastive view groups $G \in \{1, ..., L\}$.

5.1.4 Baselines. To assess the effectiveness of NLGCL, we compare it against two categories of baselines: (1) CF-based models (**MF-BPR** [30], **NGCF** [37], **LightGCN** [17], **IMP-GCN** [23], and **LayerGCN** [56]) and (2) CL-based models (**SGL** [40], **NCL** [22], **SimGCL** [51], **LightGCL** [2], **DCCF** [29], and **BIGCF** [54]).

5.2 Overall Performance (RQ1)

Table 3 reports the recommendation performance of all baselines on four public datasets. From the results, we observed that:
Observation1: Our proposed NLGCL achieves the best performance, outperforming all baselines and demonstrating its effectiveness in recommendation tasks. Specifically, NLGCL improves NDCG@10 by 4.55%, 5.69%, 2.52%, and 12.11% on Yelp, Pinterest, QB-Video, and Alibaba, respectively, compared to the strongest baseline. These results highlight NLGCL's strong capability in providing personalized recommendations.
Observation2: Across all datasets, CL-based methods generally outperform CF-based methods, highlighting the benefits of integrating contrastive learning with collaborative filtering. However, existing CL-based methods often lose crucial information and introduce noise during view construction. In contrast, NLGCL mitigates this issue by associating multiple positive samples with each node, where noise among these samples reflects other nodes with similar preferences.

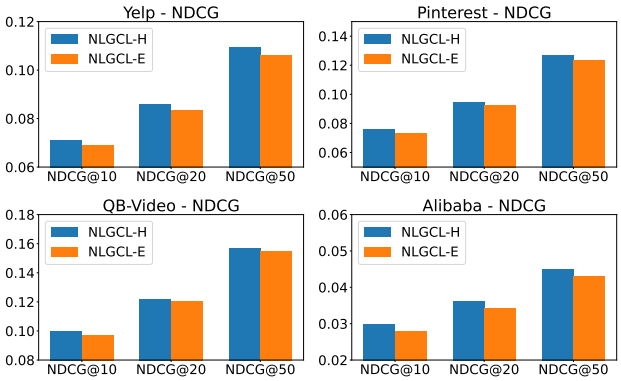

Figure 2: The influence of the scope of contrastive views.

5.3 The Influence of Different Scopes in Contrastive Views (RQ2)

To validate the effectiveness of different scopes of contrastive views, we conduct experiments on all datasets. We design the following variants: 1) NLGCL-H, which adopts the heterogeneous scope. 2) NLGCL-E, which adopts the entire scope. In Figure 2, we observed that NLGCL-H consistently outperforms NLGCL-E across all datasets. This advantage is attributable to the scope of NLGCL-H being more accurate than NLGCL-E. Specifically, there is an inevitable semantic gap between users and items. To this end, arbitrarily treating all nodes as negative samples will aggravate semantic noise in user/item representation and hurt recommendation accuracy. Moreover, NLGCL-E also requires higher computational resources than NLGCL-H, which we detail in Section 5.4.

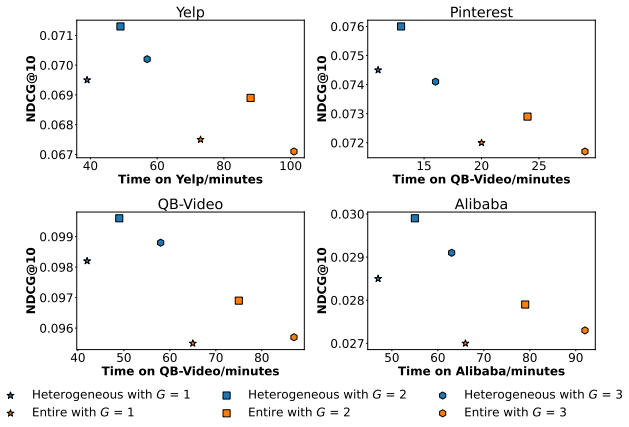

Figure 3: Efficiency study in terms of NDCG@10.

5.4 Efficiency Study (RQ3)

We provide theoretical analysis for the efficiency of our proposed NLGCL in Section 6. In this section, we further validate it with an empirical study. We conduct experiments across all CL-based baselines using four datasets. In Table 4, we present the efficiency of NLGCL and CL-based baselines in terms of average training time per epoch, the number of epochs to converge, and total training time. Our results show that NLGCL achieves competitive per-epoch training efficiency and the fastest convergence among all models. This indicates that NLGCL not only reduces computational time but also reaches optimal performance swiftly, making it ideal for scenarios with limited time or computational resources, such as rapid deployment and frequent updates. As analyzed in Section 6, our NLGCL-H and NLGCL-E further reduce training time by adjusting hyper-parameter G. Figure 3 illustrates the trade-off between performance and efficiency across different values of G. By tuning G, NLGCL adapts to diverse computational environments, balancing efficiency and effectiveness. This flexibility ensures its suitability for real-world applications, even in resource-limited scenarios, while maintaining strong performance.

5.5 Visualization Analysis (RQ4)

We validate the effectiveness of NLGCL in learning high-quality, uniformly distributed representations, thereby maximizing the benefits of CL. Specifically, we randomly sample 3,000 users from the Yelp and Pinterest datasets and project their final embeddings from the optimal model onto 2-dimensional unit vectors using t-SNE [34]. As shown in Figure 4, we plot the feature distributions and estimate their angular densities using Gaussian Kernel Density Estimation (KDE) [7]. Figure 4 demonstrates that the representations learned by NLGCL are more uniformly distributed. Consistent with prior studies [35, 36, 51, 54], such uniform distribution enhances the intrinsic quality of representations and maximizes information retention, leading to a better ability to preserve unique user preferences.

5.6 Hyper-parameter Study (RQ5)

This section investigates the sensitivity of hyper-parameters on the recommendation performance of NLGCL. The evaluation results in terms of NDCG@10 are reported in Figure 5 and Figure 6.

Performance Comparison *w.r.t.* λ_1 **and** τ**:** We analyze the hyper-parameter sensitivity for contrastive learning by varying the balancing hyper-parameter λ_1 from 10^{-6} to 10^{-4} and temperature parameter τ from 0.1 to 0.4. As shown in Figure 5, NLGCL consistently achieves the best performance with $\lambda_1 = 10^{-5}$ and $\tau = 0.2$ across all datasets, aligning with hyper-parameter settings in traditional CL.

Performance Comparison *w.r.t.* L **and** G**:** We analyze the influence of GCN layer number L and contrastive view group number G across all datasets. Figure 6 shows that $G = 2$ is optimal across all datasets, while the optimal L is 2 for Yelp and Alibaba datasets, and 3 for Pinterest and QB-Video datasets. Notably, even with $G = 1$, NLGCL maintains strong performance, consistent with the efficiency analysis in Section 5.4.

6 Efficiency Analysis

We analyze the complexity of NLGCL and compare it with LightGCN, SGL, NCL, SimGCL, LightGCL, DCCF, and BIGCF. The discussion is within a single batch since the in-batch negative sampling is a widely used trick in CL [4]. As Table 5 shows, we divide computational complexity into three components: **Encoder**, **BPR Loss**,

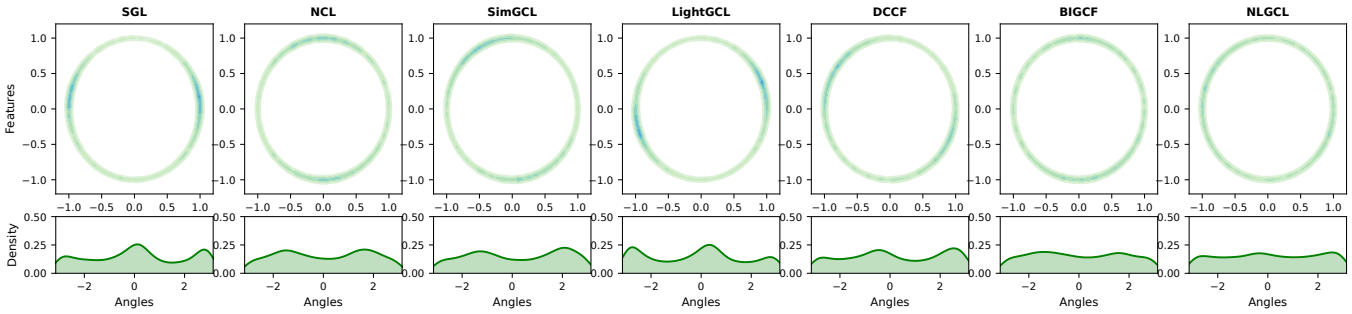

Figure 4: We follow previous work [51] to plot feature distributions with Gaussian kernel density estimation (KDE) in $\mathcal{R}^2$ (the darker the color is, the more points fall in that area.) and KDE on angles (i.e., arctan2(y, x) for each point (x,y)).

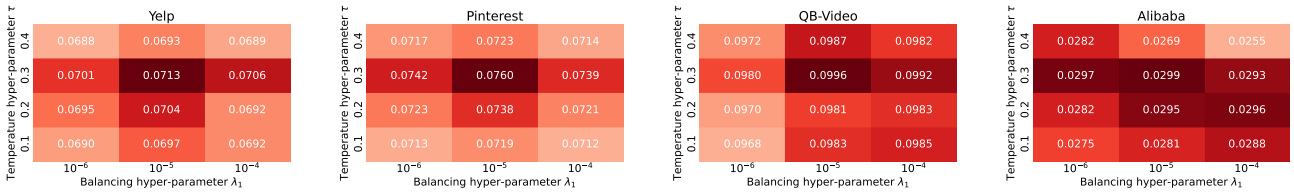

Figure 5: Performance Comparison $w.r.t.$ λ_1 and τ.

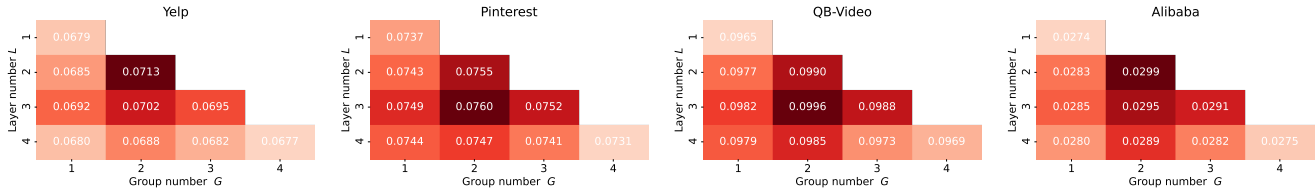

Figure 6: Performance Comparison $w.r.t.$ L and G.

Table 5: Comparison of time complexity. -H and -E denote heterogeneous and entire scopes, respectively.

	Encoder	BPR Loss	CL Loss
LightGCN	$O(2\|\mathcal{E}\|Ld)$	$O(2dB)$	-
SGL	$O(2(1+2\hat{\rho})\|\mathcal{E}\|Ld)$	$O(2dB)$	$O(2MdB)$
NCL	$O(2(\|\mathcal{E}\|+K)Ld)$	$O(2dB)$	$O(4MdB)$
SimGCL	$O(6\|\mathcal{E}\|Ld)$	$O(2dB)$	$O(2MdB)$
LightGCL	$O(2(\|\mathcal{E}\|+q\|\mathcal{V}\|)Ld)$	$O(2dB)$	$O(2MdB)$
DCCF	$O(2(\|\mathcal{E}\|+\|\mathcal{K}\|\|\mathcal{V}\|)Ld)$	$O(2dB)$	$O(6MdB)$
BIGCF	$O(2(\|\mathcal{E}\|Ld+\|\mathcal{K}\|\|\mathcal{V}\|d))$	$O(2dB)$	$O(6MdB)$
NLGCL-H	$O(2\|\mathcal{E}\|Ld)$	$O(2dB)$	$O(2GMdB)$
NLGCL-E	$O(2\|\mathcal{E}\|Ld)$	$O(2dB)$	$O(4GMdB)$

and **CL Loss**. LightGCN is the basic graph encoder, the time complexity of the process is $O(2|\mathcal{E}|Ld)$, where $|\mathcal{E}|$ is the number of edges in graph $\mathcal{G}$, M represents the node number in a batch, L is layer number, and d is the dimension of embeddings. $\hat{\rho}$ is the edge keep probability of SGL. K is the number of clusters in NCL. q is the required rank for SVD in LightGCL. $|\mathcal{K}|$ denotes the collective intent number for all user and item nodes. For our NLGCL, G is the number of groups of contrastive views. NLGCL-E requires double samples compared to NLGCL-H in CL loss. In Section 5.2 and Section 5.4, we empirically verify that NLGCL-H achieves superior results than NLGCL-E in terms of both effectiveness and efficiency. Since NLGCL-H does not construct additional contrast views through data augmentation, there is no additional cost to the Encoder beyond the backbone LightGCN. For CL Loss, it requires G times the computational cost of other CL-based models due to the multiple groups of natural contrast views. Note that we also empirically prove that (In Section 5.4): When $G = 1$, the performance of NLGCL-H has exceeded the existing baselines.

7 Proof of Theorem 1

Formal Restatement: In GNNs, the effectiveness of naturally existing contrastive views between adjacent layers $l-1$ and l diminishes as the layer index l increases. Specifically, let the information gain $\mathbf{I}(\mathbf{E}^{(l-1)};\mathbf{E}^{(l)})$ denote the mutual information between embeddings of adjacent layers. Then, monotonically decreases as l increases.

Notation: **(Mutual Information)** Define mutual information between adjacent layers as:

$$\mathbf{I}(\mathbf{E}^{(l-1)};\mathbf{E}^{(l)}) = \mathbf{H}(\mathbf{E}^{(l-1)}) - \mathbf{H}(\mathbf{E}^{(l-1)}|\mathbf{E}^{(l)}), \tag{11}$$

where $\mathbf{H}(\cdot)$ denotes the entropy function.
(Spectral Decomposition) Let $\tilde{\mathcal{A}} = \mathbf{U}\Lambda\mathbf{U}^\top$ be the eigendecomposition of $\tilde{\mathcal{A}}$ where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$

Lemma 1. ***(Entropy Reduction via Low-Pass Filtering).*** *For any layer l, the entropy of embeddings satisfies:*

$$\mathbf{H}(\mathbf{E}^{(l)}) \leq \mathbf{H}(\mathbf{E}^{(l-1)})$$

Proof: Propagation process $\mathbf{E}^{(l)} = \tilde{\mathcal{A}}\mathbf{E}^{(l-1)}$ acts as a low-pass filter in the spectral domain. Specifically, the frequency response at the k-th eigencomponent is attenuated by λ_k^l. Since $\lambda_k \leq 1$ for all k, higher-frequency components (associated with $\lambda_k < 1$) are suppressed exponentially with l, reducing the variance of $\mathbf{E}^{(l)}$. From the entropy-power inequality [10]:

$$\begin{aligned}\mathbf{H}(\mathbf{E}^{(l)}) &= \frac{1}{2}\log((2\pi e)^d|\Sigma^{(l)}|) \\ &\leq \frac{1}{2}\log((2\pi e)^d|\Sigma^{(l-1)}|) = \mathbf{H}(\mathbf{E}^{(l-1)}),\end{aligned} \tag{12}$$

where $|\Sigma^{(l)}|$ is the determinant of the covariance matrix, and d is the embedding dimension.

Corollary 1. ***(Exponential Mutual Information Decay).*** *The mutual information between adjacent layers decays as:*

$$\mathbf{I}(\mathbf{E}^{(l-1)}; \mathbf{E}^{(l)}) \propto \lambda_{\max}^{2l},$$

where $\lambda_{\max} = \max_k |\lambda_k| < 1$.

Proof: From Lemma 1, $\mathbf{H}(\mathbf{E}^{(l)})$ decreases monotonically. For a Gaussian embedding distribution, mutual information simplifies to:

$$\mathbf{I}(\mathbf{E}^{(l-1)}; \mathbf{E}^{(l)}) = \frac{1}{2}\log(\frac{|\Sigma^{(l-1)}|}{|\Sigma^{(l)}|}). \tag{13}$$

Substituting $\Sigma^{(l)} = \tilde{\mathcal{A}}^2\Sigma^{(l-1)}$ (from $\mathbf{E}^{(l)} = \tilde{\mathcal{A}}\mathbf{E}^{(l-1)}$), we derive:

$$|\Sigma^{(l)}| = |\tilde{\mathcal{A}}^2||\Sigma^{(l-1)}| = \prod_{k=1}^{n}\lambda_k^2 \cdot |\Sigma^{(l-1)}|. \tag{14}$$

Thus:

$$\mathbf{I}(\mathbf{E}^{(l-1)}; \mathbf{E}^{(l)}) = \frac{1}{2}\log(\prod_{k=1}^{n}\lambda_k^{-2}) = -\sum_{k=1}^{n}\log\lambda_k. \tag{15}$$

For dominant eigenvalues $\lambda_{\max}$, this decays as $O(\lambda_{\max}^{2l})$. Therefore, contrastive views constructed from lower layers (l = 1) maximize the signal-to-noise ratio (SNR) for contrastive learning, as they preserve higher mutual information. The SNR of contrastive pairs is proportional to $\mathbf{I}(\mathbf{E}^{(l-1)}; \mathbf{E}^{(l)})$. From Corollary 1, SNR decays exponentially with l, making lower layers preferable.

8 Related Work

8.1 GNN-based Recommendation

Graph Neural Networks (GNNs) are widely applied in recommendation systems due to their ability to capture high-order relational information in user-item bipartite graphs. NGCF [37] pioneered the use of GNNs for aggregating high-order neighborhood information, laying the foundation for subsequent advancements. GCCF [3] and LightGCN [17] enhanced encoding quality by removing non-linear transformations during propagation, which has since become a standard backbone. Further developments include IMP-GCN [23], which applies GNNs to sub-graphs, and LayerGCN [56], which incorporates residual networks into GNN architectures. FourierKAN-GCF [45] optimizes non-linear transformations using Fourier series. Beyond traditional recommendation, GNNs have been applied in multimodal [16, 38, 48, 57], social [12, 21, 27, 37], and group [5, 32, 41, 44] recommendations to leverage the structural richness of graphs. However, GNN-based methods often rely heavily on supervised signals, limiting their performance on sparse datasets.

8.2 CL-based Recommendation

Contrastive Learning (CL) addresses data sparsity in recommendation by leveraging self-supervised signals to construct contrastive views and maximize mutual information in embeddings. SGL [40] ensures consistency between views using random corruption operators, such as node/edge deletion and random walks. SimGCL [51] adds noise during graph convolution, while LightGCL [2] employs Singular Value Decomposition (SVD) to generate views. HCCF [42] integrates global information via hypergraph neural networks. NCL [22] introduces EM clustering, and DCCF [29] learns disentangled representations using self-supervised signals. BIGCF [54] explores the individuality and collectivity of user intents with tailored signals. While effective, existing CL methods incur additional computational costs and introduce irrelevant noise during contrastive view construction. Our NLGCL leverages naturally existing contrastive views within GNNs, reducing computational overhead and ensuring that the noise between contrastive views is semantically related, thereby achieving both improvements in effectiveness and efficiency.

9 Conclusion

In this paper, we propose a simple yet effective paradigm, NLGCL, which leverages naturally existing contrastive views between neighbor layers within GNNs. By treating each node and its neighbors in the next layer as positive pairs and other nodes as negatives, NLGCL avoids augmentation-based noise while preserving semantic relevance. This approach reduces irrelevant noise and eliminates the additional time and space costs of traditional contrastive learning methods. We demonstrate the superiority of NLGCL over state-of-the-art baselines in terms of both effectiveness and efficiency through theoretical analysis and empirical evaluation. By eliminating the need to construct and store additional contrastive views, NLGCL breaks the limitations of traditional GCL-based recommendation paradigms and opens new avenues for research. In the future, we plan to extend this paradigm to other domains, such as multimodal, social, and group recommendations.

Acknowledgments

This work was supported by the Hong Kong UGC General Research Fund no. 17203320 and 17209822, and the project grants from the HKU-SCF FinTech Academy.

References

[1] Markus Bayer, Marc-André Kaufhold, and Christian Reuter. 2022. A survey on data augmentation for text classification. *Comput. Surveys* 55, 7 (2022), 1–39.
[2] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *The Eleventh International Conference on Learning Representations.*

[3] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 27–34.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[5] Tong Chen, Hongzhi Yin, Jing Long, Quoc Viet Hung Nguyen, Yang Wang, and Meng Wang. 2022. Thinking inside the box: learning hypercube representations for group recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1664–1673.

[6] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: personalized outfit generation for fashion recommendation at Alibaba iFashion. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2662–2670.

[7] Yen-Chi Chen. 2017. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology* 1, 1 (2017), 161–187.

[8] Zheyu Chen, Jinfeng Xu, and Haibo Hu. 2025. Don't Lose Yourself: Boosting Multimodal Recommendation via Reducing Node-neighbor Discrepancy in Graph Convolutional Network. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.

[9] Zheyu Chen, Jinfeng Xu, Yutong Wei, and Ziyue Peng. 2025. Squeeze and Excitation: A Weighted Graph Contrastive Learning for Collaborative Filtering. *arXiv preprint arXiv:2504.04443* (2025).

[10] Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.

[11] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[12] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[13] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. 2015. Learning image and user features for recommendation in social networks. In *Proceedings of the IEEE international conference on computer vision*. 4274–4282.

[14] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[15] Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22nd international conference on world wide web*. 593–596.

[16] Zhiqiang Guo, Jianjun Li, Guohui Li, Chaoyang Wang, Si Shi, and Bin Ruan. 2024. LGMRec: Local and Global Graph Learning for Multimodal Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8454–8462.

[17] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[18] Mengyuan Jing, Yanmin Zhu, Tianzi Zang, and Ke Wang. 2023. Contrastive self-supervised learning in recommender systems: A survey. *ACM Transactions on Information Systems* 42, 2 (2023), 1–39.

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

[20] Thomas N Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[21] Zongwei Li, Lianghao Xia, and Chao Huang. 2024. Recdiff: Diffusion model for social recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 1346–1355.

[22] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM web conference 2022*. 2320–2329.

[23] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-aware message-passing GCN for recommendation. In *Proceedings of the web conference 2021*. 1296–1305.

[24] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering* 35, 1 (2021), 857–876.

[25] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2022. Graph self-supervised learning: A survey. *IEEE transactions on knowledge and data engineering* 35, 6 (2022), 5879–5900.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[27] Yuhan Quan, Jingtao Ding, Chen Gao, Lingling Yi, Depeng Jin, and Yong Li. 2023. Robust preference-guided denoising for graph based social recommendation. In *Proceedings of the ACM Web Conference 2023*. 1097–1108.

[28] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. 2021. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems* 34 (2021), 29935–29948.

[29] Xubin Ren, Lianghao Xia, Jiashu Zhao, Dawei Yin, and Chao Huang. 2023. Disentangled contrastive collaborative filtering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1137–1146.

[30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[31] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[32] Aravind Sankar, Yanhong Wu, Yuhang Wu, Wei Zhang, Hao Yang, and Hari Sundaram. 2020. Groupim: A mutual information maximization framework for neural group recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 1279–1288.

[33] Brent Smith and Greg Linden. 2017. Two decades of recommender systems at Amazon. com. *Ieee internet computing* 21, 3 (2017), 12–18.

[34] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[35] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 1816–1825.

[36] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*. PMLR, 9929–9939.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[38] Wei Wei, Chao Huang, Lianghao Xia, and Chuxu Zhang. 2023. Multi-Modal Self-Supervised Learning for Recommendation. In *Proceedings of the ACM Web Conference 2023*. 790–800.

[39] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[40] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 726–735.

[41] Xixi Wu, Yun Xiong, Yao Zhang, Yizhu Jiao, Jiawei Zhang, Yangyong Zhu, and Philip S Yu. 2023. Consrec: Learning consensus behind interactions for group recommendation. In *Proceedings of the ACM Web Conference 2023*. 240–250.

[42] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*. 70–79.

[43] Jinfeng Xu, Zheyu Chen, Jinze Li, Shuo Yang, Hewei Wang, Yijie Li, Mengran Li, Puzhen Wu, and Edith CH Ngai. 2025. MDVT: Enhancing Multimodal Recommendation with Model-Agnostic Multimodal-Driven Virtual Triplets. *arXiv preprint arXiv:2505.16665* (2025).

[44] Jinfeng Xu, Zheyu Chen, Jinze Li, Shuo Yang, Hewei Wang, and Edith CH Ngai. 2024. AlignGroup: Learning and Aligning Group Consensus with Member Preferences for Group Recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2682–2691.

[45] Jinfeng Xu, Zheyu Chen, Jinze Li, Shuo Yang, Wei Wang, Xiping Hu, and Edith C-H Ngai. 2024. FourierKAN-GCF: Fourier Kolmogorov-Arnold Network–An Effective and Efficient Feature Transformation for Graph Collaborative Filtering. *arXiv preprint arXiv:2406.01034* (2024).

[46] Jinfeng Xu, Zheyu Chen, Zixiao Ma, Jiyi Liu, and Edith CH Ngai. 2024. Improving Consumer Experience With Pre-Purify Temporal-Decay Memory-Based Collaborative Filtering Recommendation for Graduate School Application. *IEEE Transactions on Consumer Electronics* (2024).

[47] Jinfeng Xu, Zheyu Chen, Wei Wang, Xiping Hu, Sang-Wook Kim, and Edith CH Ngai. 2025. COHESION: Composite Graph Convolutional Network with Dual-Stage Fusion for Multimodal Recommendation. *arXiv preprint arXiv:2504.04452* (2025).

[48] Jinfeng Xu, Zheyu Chen, Shuo Yang, Jinze Li, Hewei Wang, and Edith CH Ngai. 2025. Mentor: multi-level self-supervised learning for multimodal recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 12908–12917.

[49] Jinfeng Xu, Zheyu Chen, Shuo Yang, Jinze Li, Wei Wang, Xiping Hu, Steven Hoi, and Edith Ngai. 2025. A Survey on Multimodal Recommender Systems: Recent Advances and Future Directions. *arXiv preprint arXiv:2502.15711* (2025).

[50] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2023. XSimGCL: Towards extremely simple graph contrastive

learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[51] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1294–1303.

[52] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, et al. 2022. Tenrec: A large-scale multipurpose benchmark dataset for recommender systems. *Advances in Neural Information Processing Systems* 35 (2022), 11480–11493.

[53] Dan Zhang, Yangliao Geng, Wenwen Gong, Zhongang Qi, Zhiyu Chen, Xing Tang, Ying Shan, Yuxiao Dong, and Jie Tang. 2024. RecDCL: Dual Contrastive Learning for Recommendation. In *Proceedings of the ACM on Web Conference 2024*. 3655–3666.

[54] Yi Zhang, Lei Sang, and Yiwen Zhang. 2024. Exploring the Individuality and Collectivity of Intents behind Interactions for Graph Collaborative Filtering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1253–1262.

[55] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, et al. 2022. RecBole 2.0: towards a more up-to-date recommendation library. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4722–4726.

[56] Xin Zhou, Donghui Lin, Yong Liu, and Chunyan Miao. 2023. Layer-refined graph convolutional networks for recommendation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1247–1259.

[57] Xin Zhou and Zhiqi Shen. 2023. A tale of two graphs: Freezing and denoising graph structures for multimodal recommendation. In *Proceedings of the 31st ACM International Conference on Multimedia*. 935–943.