

# XSimGCL: Towards Extremely Simple Graph Contrastive Learning for Recommendation

Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, Hongzhi Yin\*

**Abstract**—Contrastive learning (CL) has recently been demonstrated critical in improving recommendation performance. The fundamental idea of CL-based recommendation models is to maximize the consistency between representations learned from different graph augmentations of the user-item bipartite graph. In such a self-supervised manner, CL-based recommendation models are expected to extract general features from the raw data to tackle the data sparsity issue. Despite the effectiveness of this paradigm, we still have no clue what underlies the performance gains. In this paper, we first reveal that CL enhances recommendation through endowing the model with the ability to learn more evenly distributed user/item representations, which can implicitly alleviate the pervasive popularity bias and promote long-tail items. Meanwhile, we find that the graph augmentations, which were considered a necessity in prior studies, are relatively unreliable and less significant in CL-based recommendation. On top of these findings, we put forward an **eXtremely Simple Graph Contrastive Learning** method (**XSimGCL**) for recommendation, which discards the ineffective graph augmentations and instead employs a simple yet effective noise-based embedding augmentation to create views for CL. A comprehensive experimental study on three large and highly sparse benchmark datasets demonstrates that, though the proposed method is extremely simple, it can smoothly adjust the uniformity of learned representations and outperforms its graph augmentation-based counterparts by a large margin in both recommendation accuracy and training efficiency. The code is released at <https://github.com/Coder-Yu/SELFRec>.

**Index Terms**—Recommendation, Self-Supervised Learning, Contrastive Learning, Data Augmentation.

## 1 INTRODUCTION

Recently, a revival of contrastive learning (CL) [1], [2], [3] has swept across many fields of deep learning, leading to a series of major advances [4], [5], [6], [7], [8]. Since the ability of CL to learn general features from unlabeled raw data is a sliver bullet for addressing the data sparsity issue [9], [10], [11], it also pushes forward the frontier of recommendation. A flurry of enthusiasm on CL-based recommendation [12], [13], [14], [15], [16], [17], [18] has recently been witnessed, followed by a string of promising results. Based on these studies a paradigm of contrastive recommendation can be clearly profiled. It mainly includes two steps: first augmenting the original user-item bipartite graph with structure perturbations (e.g., edge/node dropout at a certain ratio), and then maximizing the consistency of representations learned from different graph augmentations [3] under a joint learning framework (shown in Fig. 1).

Despite the effectiveness of this paradigm, we still have no clue what underlies the performance gains. Intuitively, encouraging the agreement between related graph augmentations can learn representations invariant to slight structure perturbations and capturing the essential information of the original user-item bipartite [1], [19]. However, it is

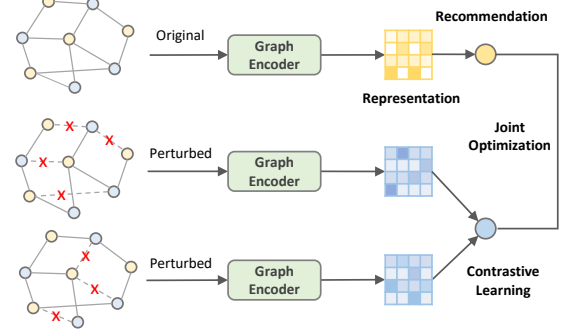


Fig. 1: Graph contrastive learning with edge dropout for recommendation.

surprising that several recent studies [17], [20], [21] have reported that the performances of their CL-based recommendation models are not sensitive to graph augmentations with different edge dropout rates, and even a large drop rate (e.g., 0.9) can somehow benefit the model. Such a finding is elusive, because a large dropout rate will result in a huge loss of the raw structural information and should have had a negative effect only. This naturally raises an intriguing and fundamental question: *Are graph augmentations really a necessity for CL-based recommendation models?*

To answer this question, we first conduct experiments with and without graph augmentations and compare the performances. The results show that when graph augmentations are detached, there is only a slight performance drop. We then investigate the representations learned by non-CL and CL-based recommendation methods. By visualizing the representations, we observe that the jointly optimized contrastive loss InfoNCE [22] is what really matters, rather

- J. Yu, X. Xia, T. Chen, and H. Yin are with the School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Queensland, Australia.  
E-mail: {jl.yu, x.xia, tong.chen, h.yin1}@uq.edu.au
- H. Nguyen is with the Institute for Integrated and Intelligent Systems, Griffith University, Gold Coast, Australia.  
E-mail: quocviet Hung1@gmail.com
- Lizhen Cui is with the School of Software, Shandong University, Jinan, China.  
E-mail: clz@sdu.edu.cn

\*Corresponding author.

than graph augmentations. Optimizing this contrastive loss always leads to more evenly distributed user/item representations regardless of graph augmentations, which implicitly alleviates the pervasive popularity bias [23] and promotes long-tail items. On the other hand, though not as effective as expected, some types of graph augmentations indeed improve the recommendation performance, which are analogous to the cherry on the cake. However, to pick out the useful ones, a long process of trial-and-error is needed. Otherwise a random selection may degrade the recommendation performance. Besides, it should be aware that repeatedly creating graph augmentations and constructing adjacency matrices bring extra expense to model training. In view of these weakness, it is sensible to substitute graph augmentations with better alternatives. A follow-up question then arises: *Are there more effective and efficient augmentation approaches?*

In our preliminary study [24], we had given an affirmative response to this question. On top of our finding that learning more evenly distributed representations is critical for boosting recommendation performance, we proposed a graph-augmentation-free CL method which makes the uniformity more controllable, and named it **SimGCL** (short for **Simple Graph Contrastive Learning**). SimGCL conforms to the paradigm presented in Fig. 1, but it discards the ineffective graph augmentations and instead adds uniform noises to the learned representations for a far more efficient representation-level data augmentation. We empirically demonstrated that this noise-based augmentation can directly regularize the embedding space towards a more even representation distribution. Meanwhile, by modulating the magnitude of noises, SimGCL can smoothly adjust the uniformity of representations. Benefitting from these characteristics, SimGCL shows superiorities over its graph augmentation-based counterparts in both recommendation accuracy and training efficiency.

However, in spite of these advantages, **the cumbersome architecture of SimGCL makes it less than perfect**. In addition to the forward/backward pass for the recommendation task, it requires two extra forward/backward passes for the contrastive task in a mini-batch (shown in Fig. 2). Actually, this is a universal problem for all the CL-based recommendation models [12], [25], [17], [26], [27] under the paradigm in Fig. 1. What makes it worse is that these methods require that all nodes in the user-item bipartite graph to be present during training, which means their computational complexity is almost triple that of conventional recommendation models. This flaw greatly limits the use of CL-based models at scale.

In order to address this issue, in this work we put forward an **eXtremely Simple Graph Contrastive Learning** method (**XSimGCL**) for recommendation. XSimGCL not only inherits SimGCL's noise-based augmentation but also drastically reduces the computational complexity by streamlining the propagation process. As shown in Fig. 2, the recommendation task and the contrastive task of XSimGCL share the forward/backward propagation in a mini-batch instead of owning separate pipelines. To be specific, both SimGCL and XSimGCL are fed with the same input: the initial embeddings and the adjacency matrix. The difference is that SimGCL contrasts two final representations learned

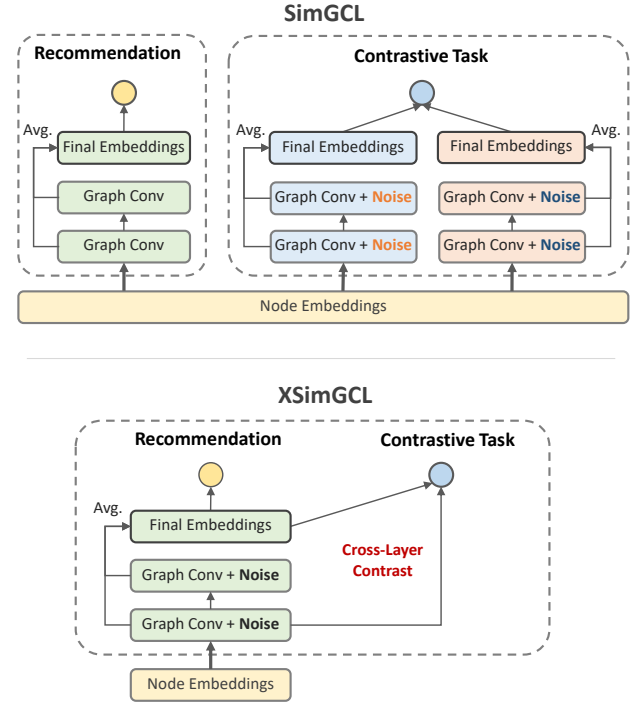


Fig. 2: The architectures of SimGCL and XSimGCL.

with different noises and relies on the ordinary representations for recommendation, whereas XSimGCL uses the same perturbed representations for both tasks, and replaces the final-layer contrast in SimGCL with the cross-layer contrast. With this new design, XSimGCL is nearly as lightweight as the conventional recommendation model LightGCN [28]. And to top it all, XSimGCL even outperforms SimGCL with its simpler architecture.

Overall, as an extension to our conference paper [24], this work makes the following contributions:

- We reveal that CL enhances graph recommendation models by learning more evenly distributed representations where the InfoNCE loss is far more important than graph augmentations.
- We propose a simple yet effective noise-based augmentation approach, which can smoothly adjust the uniformity of the representation distribution through contrastive learning.
- We put forward a novel CL-based recommendation model XSimGCL which is even more effective and efficient than its predecessor SimGCL.
- We conduct a comprehensive experimental study on three large and highly sparse benchmark datasets (two of which were not used in our preliminary study) to demonstrate that XSimGCL is an ideal alternative of its graph augmentation-based counterparts.

The rest of this paper is organized as follows. Section 2 investigates the necessity of graph augmentations in the contrastive recommendation and explores how CL enhances recommendation. Section 3 proposes the noise-based augmentation approach and the CL-based recommendation model XSimGCL. The experimental study is presented in Section 4. Section 5 provides a brief review of the related literature. Finally, we conclude this work in Section 6.

## 2 REVISITING GRAPH CONTRASTIVE LEARNING FOR RECOMMENDATION

### 2.1 Contrastive Recommendation with Graph Augmentations

Generally, data augmentations are the prerequisite for CL-based recommendation models [15], [12], [13], [25]. In this section, we investigate the widely used dropout-based augmentations on graphs [12], [4]. They assume that the learned representations which are invariant to partial structure perturbations are high-quality. We target a representative state-of-the-art CL-based recommendation model SGL [12], which performs the node/edge dropout to augment the user-item graph. The joint learning scheme in SGL is formulated as:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda \mathcal{L}_{cl}, \quad (1)$$

which consists of the recommendation loss  $\mathcal{L}_{rec}$  and the contrastive loss  $\mathcal{L}_{cl}$ . Since the goal of SGL is to recommend items, the CL task plays an auxiliary role and its effect is modulated by a hyperparameter  $\lambda$ . As for the concrete forms of these two losses, SGL adopts the standard BPR loss [29] for recommendation and the InfoNCE loss [22] for CL. The standard BPR loss is defined as:

$$\mathcal{L}_{rec} = - \sum_{(u,i) \in \mathcal{B}} \log \left( \sigma(\mathbf{e}_u^\top \mathbf{e}_i - \mathbf{e}_u^\top \mathbf{e}_j) \right), \quad (2)$$

where  $\sigma$  is the sigmoid function,  $\mathbf{e}_u$  is the user representation,  $\mathbf{e}_i$  is the representation of an item that user  $u$  has interacted with,  $\mathbf{e}_j$  is the representation of a randomly sampled item, and  $\mathcal{B}$  is a mini-batch. The InfoNCE loss [22] is formulated as:

$$\mathcal{L}_{cl} = \sum_{i \in \mathcal{B}} -\log \frac{\exp(\mathbf{z}_i'^\top \mathbf{z}_i''/\tau)}{\sum_{j \in \mathcal{B}} \exp(\mathbf{z}_i'^\top \mathbf{z}_j''/\tau)}, \quad (3)$$

where  $i$  and  $j$  are users/items in  $\mathcal{B}$ ,  $\mathbf{z}_i'$  and  $\mathbf{z}_i''$  are  $L_2$  normalized representations learned from two different dropout-based graph augmentations (namely  $\mathbf{z}_i' = \frac{\mathbf{e}_i'}{\|\mathbf{e}_i'\|_2}$ ), and  $\tau > 0$  (e.g., 0.2) is the temperature which controls the strength of penalties on hard negative samples. The InfoNCE loss encourages the consistency between  $\mathbf{z}_i'$  and  $\mathbf{z}_i''$  which are the positive sample of each other, whilst minimizing the agreement between  $\mathbf{z}_i'$  and  $\mathbf{z}_j''$ , which are the negative samples of each other. Optimizing the InfoNCE loss is actually maximizing a tight lower bound of mutual information.

To learn representations from the user-item graph, SGL employs LightGCN [28] as its encoder, whose message passing process is defined as:

$$\mathbf{E} = \frac{1}{1+L} (\mathbf{E}^{(0)} + \mathbf{A}\mathbf{E}^{(0)} + \dots + \mathbf{A}^L \mathbf{E}^{(0)}), \quad (4)$$

where  $\mathbf{E}^{(0)} \in \mathbb{R}^{|N| \times d}$  is the node embeddings to be learned,  $\mathbf{E}$  is the initial final representations for prediction,  $|N|$  is the number of nodes,  $L$  is the number of layers, and  $\mathbf{A} \in \mathbb{R}^{|N| \times |N|}$  is the normalized undirected adjacency matrix without self-connection. By replacing  $\mathbf{A}$  with the adjacency matrix of the corrupted graph augmentations  $\tilde{\mathbf{A}}$ ,  $\mathbf{z}'$  and  $\mathbf{z}''$  can be learned via Eq. (4). It should be noted that in every epoch,  $\tilde{\mathbf{A}}$  is reconstructed. For the sake of brevity, here we just present the core ingredients of SGL and LightGCN. More details can be found in the original papers [12], [28].

TABLE 1: Performance comparison of different SGL variants.

Method	Yelp2018		Kindle		iFashion	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN	0.0639	0.0525	0.2053	0.1315	0.0955	0.0461
SGL-ND	0.0644	0.0528	0.2069	0.1328	0.1032	0.0498
SGL-ED	<b>0.0675</b>	<b>0.0555</b>	0.2090	<b>0.1352</b>	0.1093	<b>0.0531</b>
SGL-RW	0.0667	0.0547	<b>0.2105</b>	0.1351	<b>0.1095</b>	<b>0.0531</b>
SGL-WA	0.0671	0.0550	0.2084	0.1347	0.1065	0.0519

### 2.2 Necessity of Graph Augmentations

The phenomenon reported in [17], [20], [21] that even very sparse graph augmentations can somehow benefit the recommendation model suggest that CL-based recommendation may work in a way different from what we reckon. To demystify how CL enhances recommendation, we first investigate the necessity of the graph augmentation in SGL. In the original paper of SGL [12], three variants are proposed including SGL-ND (-ND denotes node dropout), SGL-ED (-ED denotes edge dropout), and SGL-RW (-RW denotes random walk, i.e., multi-layer edge dropout). For a control group, we construct a new variant of SGL, termed **SGL-WA** (-WA stands for w/o augmentation) in which the CL loss is defined as:

$$\mathcal{L}_{cl} = \sum_{i \in \mathcal{B}} -\log \frac{\exp(1/\tau)}{\sum_{j \in \mathcal{B}} \exp(\mathbf{z}_i'^\top \mathbf{z}_j/\tau)}. \quad (5)$$

Because no augmentations are used in SGL-WA, we have  $\mathbf{z}_i' = \mathbf{z}_i'' = \mathbf{z}_i$ . The performance comparison is conducted on three benchmark datasets: *Yelp2018* [28], *Amazon-Kindle* [30] and *Alibaba-iFashion* [12]. A 3-layer setting is adopted and the hyperparameters are tuned according to the original paper of SGL (more experimental settings can be found in Section 4.1). The results are presented in Table 1 where the highest values are marked in bold type.

As can be observed, all the graph augmentation-based variants of SGL outperform LightGCN, which demonstrates the effectiveness of CL. However, to our surprise SGL-WA is also competitive. Its performance is on par with that of SGL-ED and SGL-RW and is even better than that of SGL-ND on all the datasets. Given these results, we can draw two conclusions: (1) graph augmentations indeed work but they are not as effective as expected; the large proportion of performance gains derive from the contrastive loss InfoNCE and this can explain why even very sparse graph augmentations seem to be informative in [17], [20], [21]; (2) not all graph augmentations have a positive impact; a long process of trial-and-error is required to pick out the useful ones. As for (2), the possible reason could be that some graph augmentations highly distort the original graph. For example, the node dropout is very likely to drop the key nodes (e.g., hub) and their associated edges and hence breaks the correlated subgraphs into disconnected pieces. Such graph augmentations share little learnable invariance with the original graph and therefore it is unreasonable to encourage the consistency between them. By contrast, the edge dropout is at a lower risk to largely perturb the original graph so that SGL-ED/RW can hold a slim advantage over SGL-WA. However, in view of the expense of regular



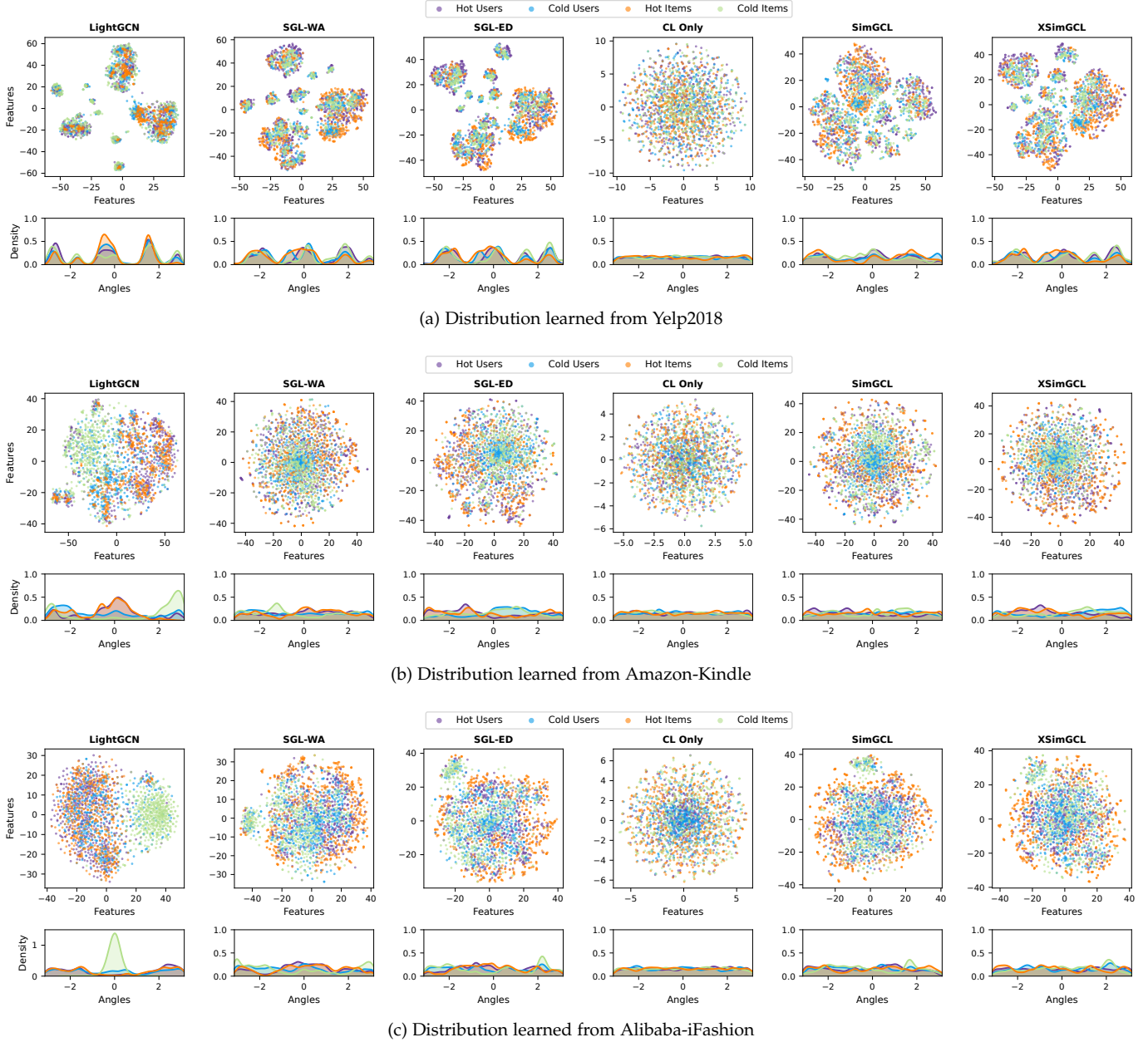


Fig. 3: The distribution of representations learned from three datasets. The top of each figure plots the learned 2D features and the bottom of each figure plots the Gaussian kernel density estimation of  $\text{atan2}(y, x)$  for each point  $(x, y) \in \mathcal{S}^1$

reconstruction of the adjacency matrices during training, it is sensible to search for better alternatives.

### 2.3 Uniformity Is What Really Matters

The last section has revealed that the contrastive loss InfoNCE is the key. However, we still have no idea how it operates. The previous research [31] on the visual representation learning has identified that pre-training with the InfoNCE loss intensifies two properties: *alignment* of features from positive pairs, and *uniformity* of the normalized feature distribution on the unit hypersphere. It is unclear if the CL-based recommendation methods exhibit similar patterns under a joint learning setting. Since in recommendation the goal of  $\mathcal{L}_{rec}$  is to align the interacted user-item pair, here we focus on investigating the uniformity.

In our preliminary study [24], we have displayed the distribution of 2,000 randomly sampled users after optimizing the InfoNCE loss. For a thorough understanding, in this version we sample both users and items. Specifically, we first rank users and items according to their popularity. Then 500 hot items are randomly sampled from the item group with the top 5% interactions; the other 500 cold items are randomly sampled from the group with the bottom 80% interactions, and so are the users. Afterwards, we map the learned representations to 2-dimensional space with t-SNE [32]. All the representations are collected when the models reach their best. Then we plot the 2D feature distributions in Fig. 3. For a clearer presentation, the Gaussian kernel density estimation [33] of  $\text{arctan}(\text{feature}_y/\text{feature}_x)$  on the unit hypersphere  $\mathcal{S}^1$  is also visualized.

From Fig. 3 an obvious contrast between the features/density estimations learned by LightGCN and CL-based recommendation models can be observed. In the leftmost column, LightGCN learns highly clustered features and the density curves are with steep rises and falls. Besides, it is easy to notice that the hot users and hot items have similar distributions and the cold users also cling to the hot items; only a small number of users are scattered among the cold items. Technically, this is a biased pattern that will lead the model to continually expose hot items to most users and generate run-of-the-mill recommendations. We think that two issues may cause this biased distribution. One is that in recommender systems a fraction of items often account for most interactions [34], and the other is the notorious over-smoothing problem [35] which makes embeddings become locally similar and hence aggravates the Matthew Effect. By contrast, in the second and the third columns, the features learned by SGL variants are more evenly distributed, and the density estimation curves are less sharp, regardless of if the graph augmentations are applied. For reference, in the forth column we plot the features learned only by optimizing the InfoNCE loss in SGL-ED. Without the effect of  $\mathcal{L}_{rec}$ , the features are almost subject to uniform distributions. The following inference provides a theoretical justification for this pattern. By rewriting Eq. (3) we can derive,

$$\mathcal{L}_{cl} = \sum_{i \in \mathcal{B}} \left( -\mathbf{z}_i'^T \mathbf{z}_i'' / \tau + \log \sum_{j \in \mathcal{B}} \exp(\mathbf{z}_i'^T \mathbf{z}_j'' / \tau) \right). \quad (6)$$

When the representations of different augmentations of the same node are perfectly aligned (SGL-WA is analogous to this case), we have

$$\mathcal{L}_{cl} = \sum_{i \in \mathcal{B}} \left( -1/\tau + \log \left( \sum_{j \in \mathcal{B}/\{i\}} \exp(\mathbf{z}_i'^T \mathbf{z}_j'' / \tau) + \exp(1/\tau) \right) \right). \quad (7)$$

Since  $1/\tau$  is a constant, optimizing the CL loss is actually towards minimizing the cosine similarity between different node representations, which will push different nodes away from each other.

By aligning the results in Table 1 with the distributions in Fig. 3, it is natural to speculate that the increased uniformity of the learned distribution is what really matters to the performance gains. It implicitly alleviates the popularity bias and promotes the long-tail items (discussed in section 4.2) because more evenly distributed representations can preserve the intrinsic characteristics of nodes and improve the generalization ability. This can also justifies the unexpected remarkable performance of SGL-WA. Finally, it also should be noted that, a positive correlation between the uniformity and the performance only holds in a limited scope. The excessive pursuit to the uniformity will weaken the effect of the recommendation loss to align interacted pairs and similar users/items, and hence degrades the recommendation performance.

### 3 TOWARDS EXTREMELY SIMPLE GRAPH CONTRASTIVE LEARNING FOR RECOMMENDATION

In this section we propose a substitute of graph augmentations and develop a lightweight architecture for CL-based recommendation.

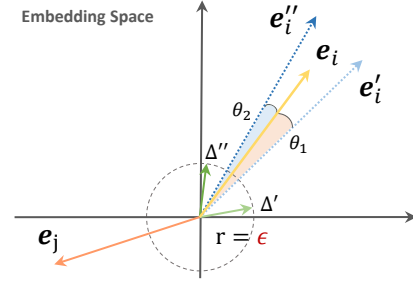


Fig. 4: An illustration of the proposed random noise-based data augmentation.

#### 3.1 Noise-Based Augmentation

Based on the findings above, we speculate that by adjusting the uniformity of the learned representation in a certain scope, contrastive recommendation models can reach a better performance. Since manipulating the graph structure for controllable uniformity is intractable and time-consuming, we shift attention to the embedding space. Inspired by the adversarial examples [36] which are constructed through adding imperceptible perturbation to the images, we propose to directly add random noises to the representation for an efficient augmentation.

Formally, given a node  $i$  and its representation  $\mathbf{e}_i$  in the  $d$ -dimensional embedding space, we can implement the following representation-level augmentation:

$$\mathbf{e}'_i = \mathbf{e}_i + \Delta'_i, \quad \mathbf{e}''_i = \mathbf{e}_i + \Delta''_i, \quad (8)$$

where the added noise vectors  $\Delta'_i$  and  $\Delta''_i$  are subject to  $\|\Delta\|_2 = \epsilon$  and  $\epsilon$  is a small constant. Technically, this constraint of magnitude makes  $\Delta$  numerically equivalent to points on a hypersphere with the radius  $\epsilon$ . Besides, it is required that:

$$\Delta = X \odot \text{sign}(\mathbf{e}_i), \quad X \in \mathbb{R}^d \sim U(0, 1), \quad (9)$$

which forces  $\mathbf{e}_i$ ,  $\Delta'$  and  $\Delta''$  to be in the same hyperoctant, so that adding the noises to  $\mathbf{e}_i$  will not result in a large deviation and construct less informative augmentations of  $\mathbf{e}_i$ . Geometrically, by adding the scaled noise vectors to  $\mathbf{e}_i$ , we can rotate it by two small angles ( $\theta_1$  and  $\theta_2$  shown in Fig. 4). Each rotation corresponds to a deviation of  $\mathbf{e}_i$ , and generates an augmented representation ( $\mathbf{e}'_i$  and  $\mathbf{e}''_i$ ). Since the rotation is small enough, the augmented representation retains most information of the original representation and meanwhile also brings some difference. Particularly, we also hope the learned representations can spread out in the entire embedding space so as to fully utilize the expression power of the space. Zhang *et al.* [37] proved that uniform distribution has such a property. We then choose to generate the noises from uniform distribution. Though it is technically difficult to make the learned distribution approximate uniform distribution in this way, it can statistically bring a hint of uniformity to the augmentation.

#### 3.2 Proposed Contrastive Recommendation Model

##### 3.2.1 A Review of SimGCL

Before presenting XSimGCL, we first briefly review SimGCL proposed in our conference paper [24], which will help understand the new contributions.

As shown in Fig. 2, SimGCL follows the paradigm of graph CL-based recommendation portrayed in Fig. 1. It consists of three encoders: one is for the recommendation task and the other two are for the contrastive task. SimGCL likewise adopts LightGCN as the backbone to learn graph representations. Since LightGCN is network-parameter-free, the input user/item embeddings are the only parameters to be learned. The ordinary graph encoding is used for recommendation which follows Eq. (4) to propagate node information. Meanwhile, in the other two encoders SimGCL employs the proposed noise-based augmentation approach, and adds different uniform random noises to the aggregated embeddings at each layer to obtain perturbed representations. This noise-involved representation learning can be formulated as:

$$\mathbf{E}' = \frac{1}{L} \left( (\mathbf{A}\mathbf{E}^{(0)} + \Delta^{(1)}) + (\mathbf{A}(\mathbf{A}\mathbf{E}^{(0)} + \Delta^{(1)}) + \Delta^{(2)}) + \dots + (\mathbf{A}^L \mathbf{E}^{(0)} + \mathbf{A}^{L-1} \Delta^{(1)} + \dots + \mathbf{A} \Delta^{(L-1)} + \Delta^{(L)}) \right) \quad (10)$$

Note that we skip the input embedding  $\mathbf{E}^{(0)}$  in all the three encoders when calculating the final representations, because we experimentally find that CL cannot consistently improve non-aggregation-based models and skipping it will lead to a slight performance improvement. Finally, we substitute the learned representations into the joint loss presented in Eq. (1) then use Adam to optimize it.

### 3.2.2 XSimGCL - Simpler Than Simple

Compared with SGL, SimGCL is much simpler because the constant graph augmentation is no longer required. However, the cumbersome architecture of SimGCL makes it less than perfect. For each computation, it requires three forward/backward passes to obtain the loss and then back-propagate the error to update the input node embeddings. Though it seems a convention to separate the pipelines of the recommendation task and the contrastive task in CL-based recommender systems [12], [24], [27], [25], we question the necessity of this architecture.

As suggested by [38], there is a sweet spot when utilizing CL where the mutual information between correlated views is neither too high nor too low. However, in the architecture of SimGCL, given a pair of views of the same node, the mutual information between two them could always be very high since the two corresponding embeddings both contain information from  $L$  hops of neighbors. Due to the minor difference between them, contrasting them with each other may be less effective in learning general features. In fact, this is also a universal problem for many CL-based recommendation models under the paradigm in Fig. 1. It is natural to think what if we contrast different layer embeddings? They share some common information but differ in aggregated neighbors and added noises, which conform to the sweet spot theory. Besides, considering that the magnitude of added noises is small enough, we can directly use the perturbed representations for the recommendation task. The noises are analogous to the widely used dropout trick and are only applied in training. In the test phase, the model switches to the ordinary mode without noises.

Benefitting from this design of cross-layer contrast, we can streamline the architecture of SimGCL by merging its

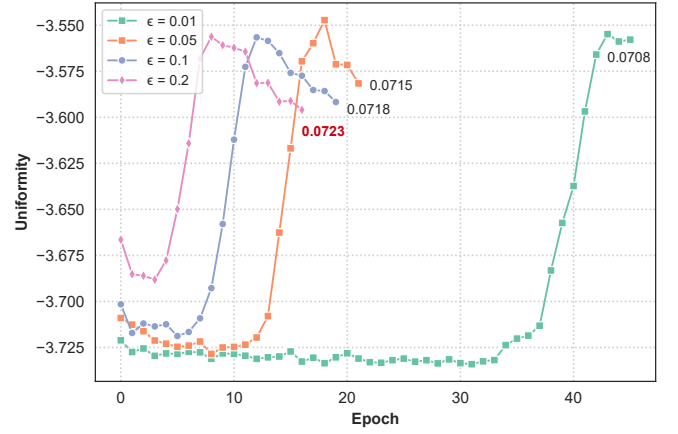


Fig. 5: Trends of uniformity with different  $\epsilon$ . Lower values on the y-axis are better. We present the Recall@20 values of XSimGCL with different  $\epsilon$  when it reaches convergence.

encoding processes. As a result, the new architecture has only one-time forward/backward pass in a mini-batch computation. We name this new method **XSimGCL**, which is short for eXtremely Simple Graph Contrastive Learning. We illustrate it in Fig. 2. The perturbed representation learning of XSimGCL is as the same as that of SimGCL. The joint loss of XSimGCL is formulated as:

$$\mathcal{L} = - \sum_{(u,i) \in \mathcal{B}} \log \left( \sigma(\mathbf{e}_u^T \mathbf{e}_i' - \mathbf{e}_u^T \mathbf{e}_j') \right) + \lambda \sum_{i \in \mathcal{B}} - \log \frac{\exp(\mathbf{z}_i^T \mathbf{z}_{i^*}^* / \tau)}{\sum_{j \in \mathcal{B}} \exp(\mathbf{z}_i^T \mathbf{z}_j^* / \tau)}, \quad (11)$$

where  $i^*$  denotes the layer to be contrasted with the final layer. Contrasting two intermediate layers is optional, but the experiments in section 4.3 show that involving the final layer leads to the optimal performance.

### 3.2.3 Adjusting Uniformity Through Changing $\epsilon$

In XSimGCL, we can explicitly control how far the augmented representations deviate from the original through changing the value of  $\epsilon$ . Intuitively, a larger  $\epsilon$  will result in a more uniform representation distribution. This is because when optimizing the contrastive loss, the added noises are also propagated as part of the gradients. As the noises are sampled from uniform distribution, the original representation is roughly regularized towards higher uniformity. We conduct the following experiment to demonstrate it.

According to [31], the logarithm of the average pairwise Gaussian potential (a.k.a. the Radial Basis Function (RBF) kernel) can well measure the uniformity of representations, which is defined as:

$$\mathcal{L}_{\text{uniform}}(f) = \log \mathbb{E}_{\substack{i, i, d \\ u, v \sim p_{\text{node}}}} e^{-2\|f(u) - f(v)\|_2^2}. \quad (12)$$

where  $f(u)$  outputs the  $L_2$  normalized embedding of  $u$ . We choose the popular items (with more than 200 interactions) and randomly sample 5,000 users in the dataset of Yelp2018 to form the user-item pairs, and then measure the uniformity of representations learned by XSimGCL with Eq. (12). We fix  $\lambda = 0.2$  and use a 3-layer setting, and



then tune  $\epsilon$  to observe how the uniformity changes. The uniformity is checked after every epoch until XSimGCL reaches convergence.

As clearly shown in Fig. 5, similar trends are observed on all the curves. At the beginning, all curves have highly evenly-distributed representations because we use Xavier initialization, which is a special uniform distribution, to initialize the input embeddings. With the training proceeding, the uniformity declines due to the effect of  $\mathcal{L}_{rec}$ . After reaching the peak the uniformity increases again till convergence. It is also obvious that with the increase of  $\epsilon$ , XSimGCL tends to learn a more even representation distribution. Meanwhile, the better performance is achieved, which evidences our claim that more uniformity brings a better performance in scope. Besides, we notice that there is a correlation between the convergence speed and the magnitude of noises, which will be discussed in section 4.3.

### 3.3 Complexity

In this section, we analyze the theoretical time complexity of XSimGCL and compare it with LightGCN, SGL-ED and its predecessor SimGCL. The discussion is within the scope of a single batch since the in-batch negative sampling is a widely used trick in CL [5]. Here we let  $|A|$  be the edge number in the user-item bipartite graph,  $d$  be the embedding dimension,  $B$  denote the batch size,  $M$  represent the node number in a batch,  $L$  be the layer number, and  $\rho$  denote the edge keep rate in SGL-ED. We can derive:

TABLE 2: The comparison of time complexity

	LightGCN	SGL-ED	SimGCL	XSimGCL
Adjacency Matrix	$\mathcal{O}(2 A )$	$\mathcal{O}(2 A +4\rho A )$	$\mathcal{O}(2 A )$	$\mathcal{O}(2 A )$
Graph Encoding	$\mathcal{O}(2 A Ld)$	$\mathcal{O}((2+4\rho) A Ld)$	$\mathcal{O}(6 A Ld)$	$\mathcal{O}(2 A Ld)$
Prediction	$\mathcal{O}(2Bd)$	$\mathcal{O}(2Bd)$	$\mathcal{O}(2Bd)$	$\mathcal{O}(2Bd)$
Contrast	-	$\mathcal{O}(BMd)$	$\mathcal{O}(BMd)$	$\mathcal{O}(BMd)$

- Since LightGCN, SimGCL and XSimGCL do not need graph augmentations, they only construct the normalized adjacency matrix which has  $2|A|$  non-zero elements. For SGL-ED, two graph augmentations are used and each has  $2\rho|A|$  non-zero elements in their adjacency matrices.
- In the phase of graph encoding, a three-encoder architecture is adopted in both SGL-ED and SimGCL to learn two different augmentations, so the encoding expense of SGL-ED and SimGCL are almost three times that of LightGCN. In contrast, the encoding expense of XSimGCL is as the same as that of LightGCN.
- As for the prediction, all methods are trained with the BPR loss and each batch contains  $B$  interactions, so they have exactly the same time cost in this regard.
- The computational cost of CL comes from the contrast between the positive/negative samples, which are  $\mathcal{O}(Bd)$  and  $\mathcal{O}(BMd)$ , respectively, because each node regards the views of itself as the positives and views of other nodes as the negatives. For the sake of brevity, we mark it as  $\mathcal{O}(BMd)$  since  $M \gg 1$ .

In these four models, SGL-ED and SimGCL are obviously the two with the highest computation costs. SimGCL needs more time in the graph encoding but SGL-ED requires

constant graph augmentations. Since this part is usually performed on CPUs, which brings SGL-ED more expense of time in practice. By comparison, XSimGCL needs neither graph augmentations nor extra encoders. Without considering the computation for the contrastive task, XSimGCL is theoretically as lightweight as LightGCN and only spends one-third of SimGCL's training expense in graph encoding. When the actual number of epochs for training is considered, XSimGCL will show more efficiency beyond what we can observe from this theoretical analysis.

TABLE 3: Dataset Statistics

Dataset	#User	#Item	#Feedback	Density
Yelp2018	31,668	38,048	1,561,406	0.13%
Amazon-Kindle	138,333	98,572	1,909,965	0.014%
Alibaba-iFashion	300,000	81,614	1,607,813	0.007%

## 4 EXPERIMENTS

### 4.1 Experimental Settings

**Datasets.** For reliable and convincing results, we conduct experiments on three public large-scale datasets: Yelp2018 [28], Amazon-kindle [12] and Alibaba-iFashion [12] to evaluate XSimGCL/SimGCL. The statistics of these datasets are presented in Table 3. We split the datasets into three parts (training set, validation set, and test set) with a 7:1:2 ratio. Following [12], [28], we first search the best hyperparameters on the validation set, and then we merge the training set and the validation set to train the model and evaluate it on the test set where the relevancy-based metric Recall@20 and the ranking-aware metric NDCG@20 are used. For a rigorous and unbiased evaluation, the reported result are the average values of 5 runs, with all the items being ranked.

**Baselines.** Besides LightGCN and the SGL variants, the following recent data augmentation-based/CL-based recommendation models are compared.

- **DNN+SSL** [27] is a recent DNN-based recommendation method which adopts the similar architecture in Fig. 1, and conducts feature masking for CL.
- **BUIR** [21] has a two-branch architecture which consists of a target network and an online network, and only uses positive examples for self-supervised recommendation.
- **MixGCL** [39] designs the hop mixing technique to synthesize hard negatives for graph collaborative filtering by embedding interpolation.
- **NCL** [18] is a very recent contrastive model which designs a prototypical contrastive objective to capture the correlations between a user/item and its context.

**Hyperparameters.** For a fair comparison, we referred to the best hyperparameter settings reported in the original papers of the baselines and then fine-tuned them with the grid search. As for the general settings, we create the user and item embeddings with the Xavier initialization of dimension 64; we use Adam to optimize all the models with the learning rate 0.001; the  $L_2$  regularization coefficient  $10^{-4}$  and the batch size 2048 are used, which are common in many papers [28], [12], [40]. In SimGCL, XSimGCL and SGL, we empirically let the temperature  $\tau = 0.2$  because this value is often reported a great choice in papers on CL [12],

TABLE 4: Performance Comparison for different CL methods on three benchmarks.

Method	Yelp2018		Amazon-Kindle		Alibaba-iFashion		
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	
1-Layer	LightGCN	0.0631	0.0515	0.1871	0.1186	0.0845	0.0390
	SGL-ND	0.0643 (+1.9%)	0.0529 (+2.7%)	0.1880 (+0.5%)	0.1192 (+0.5%)	0.0896 (+6.0%)	0.0432(+10.8%)
	SGL-ED	0.0637 (+1.0%)	0.0526 (+2.1%)	0.1936 (+3.5%)	0.1231 (+3.8%)	0.0932 (+10.3%)	0.0447 (+14.6%)
	SGL-RW	0.0637 (+1.0%)	0.0526 (+2.1%)	0.1936 (+3.5%)	0.1231 (+3.8%)	0.0932 (+10.3%)	0.0447 (+14.6%)
	SGL-WA	0.0628 (-0.4%)	0.0525 (+1.9%)	0.1918 (+2.5%)	0.1221 (+2.9%)	0.0913 (+8.0%)	0.0440 (12.8%)
	<b>SimGCL</b>	<u>0.0689 (+9.2%)</u>	<u>0.0572 (+11.1%)</u>	<b>0.2087 (+11.5%)</b>	<b>0.1361 (+14.8%)</b>	<u>0.1036 (+22.6%)</u>	<u>0.0505 (+29.5%)</u>
	<b>XSimGCL</b>	<b>0.0692 (+9.7%)</b>	<b>0.0582 (+13.0%)</b>	<u>0.2071 (+10.7%)</u>	<u>0.1339 (+12.9%)</u>	<b>0.1069 (+26.5%)</b>	<b>0.0527 (+35.1%)</b>
2-Layer	LightGCN	0.0622	0.0504	0.2033	0.1284	0.1053	0.0505
	SGL-ND	0.0658 (+5.8%)	0.0538 (+6.7%)	0.2020 (-0.6%)	0.1307 (+1.8%)	0.0993 (-5.7%)	0.0484 (-4.2%)
	SGL-ED	0.0668 (+7.4%)	0.0549 (+8.9%)	0.2084 (+2.5%)	0.1341 (+4.4%)	0.1062 (+0.8%)	0.0514 (+1.8%)
	SGL-RW	0.0644 (+3.5%)	0.0530 (+5.2%)	0.2088 (+2.7%)	0.1345 (+4.8%)	0.1053 (+0.0%)	0.0512 (+1.4%)
	SGL-WA	0.0653 (+5.0%)	0.0544 (+7.9%)	0.2068 (+1.7%)	0.1330 (+3.6%)	0.1028 (-2.4%)	0.0501 (-0.8%)
	<b>SimGCL</b>	<u>0.0719 (+15.6%)</u>	<u>0.0601 (+19.2%)</u>	<u>0.2071 (+1.9%)</u>	<u>0.1341 (+4.4%)</u>	<u>0.1119 (+6.3%)</u>	<u>0.0548 (+8.5%)</u>
	<b>XSimGCL</b>	<b>0.0722 (+16.1%)</b>	<b>0.0604 (+19.8%)</b>	<b>0.2114 (+4.0%)</b>	<b>0.1382 (+7.6%)</b>	<b>0.1143 (+8.5%)</b>	<b>0.0559 (+10.7%)</b>
3-Layer	LightGCN	0.0639	0.0525	0.2057	0.1315	0.0955	0.0461
	SGL-ND	0.0644 (+0.8%)	0.0528 (+0.6%)	0.2069 (+0.6%)	0.1328 (+1.0%)	0.1032 (+8.1%)	0.0498 (+8.0%)
	SGL-ED	0.0675 (+5.6%)	0.0555 (+5.7%)	0.2090 (+1.6%)	0.1352 (+2.8%)	0.1093 (+14.5%)	0.0531 (+15.2%)
	SGL-RW	0.0667 (+4.4%)	0.0547 (+4.5%)	0.2105 (+2.3%)	0.1351 (+2.7%)	0.1095 (+14.7%)	0.0531 (+15.2%)
	SGL-WA	0.0671 (+5.0%)	0.0550 (+4.8%)	<u>0.2084 (+1.3%)</u>	0.1347 (+2.4%)	0.1065 (+11.5%)	0.0519 (+12.6%)
	<b>SimGCL</b>	<u>0.0721 (+12.8%)</u>	<u>0.0601 (+14.5%)</u>	<u>0.2104 (+2.3%)</u>	<u>0.1374 (+4.5%)</u>	<u>0.1151 (+20.5%)</u>	<u>0.0567 (+23.0%)</u>
	<b>XSimGCL</b>	<b>0.0723 (+13.1%)</b>	<b>0.0604 (+15.0%)</b>	<b>0.2147 (+4.4%)</b>	<b>0.1415 (+7.6%)</b>	<b>0.1196 (+25.2%)</b>	<b>0.0586 (27.1%)</b>

[31]. An exception is that we let  $\tau = 0.15$  for XSimGCL on Yelp2018, which brings a slightly better performance. Note that although the paper of SGL [12] uses Yelp2018 and Alibaba-iFashion as well, we cannot reproduce their results on Alibaba-iFashion with their given hyperparameters under the same experimental setting. So we re-search the hyperparameters of SGL and choose to present our results on this dataset in Table 4.

TABLE 5: The best hyperparameters of compared methods.

Dataset	Yelp2018	Amazon-Kindle	Alibaba-iFashion
SGL	$\lambda=0.1, \rho=0.1$	$\lambda=0.05, \rho=0.1$	$\lambda=0.05, \rho=0.2$
SimGCL	$\lambda=0.5, \epsilon=0.1$	$\lambda=0.1, \epsilon=0.1$	$\lambda=0.05, \epsilon=0.1$
XSimGCL	$\lambda=0.2, \epsilon=0.2, l^*=1$	$\lambda=0.2, \epsilon=0.1, l^*=1$	$\lambda=0.05, \epsilon=0.05, l^*=3$

## 4.2 SGL vs. XSimGCL: From A Comprehensive Perspective

In this part, we compare XSimGCL with SGL in a comprehensive way. The experiments focus on three important aspects: recommendation performance, training time, and the ability to promote long-tail items.

### 4.2.1 Performance Comparison

We first show the performances of SGL and XSimGCL/SimGCL with different layers. The best hyperparameters of them are provided in Table 5 for an easy reproduction of our results. The figures of the best performance are presented in bold and the runner-ups are presented with underline. The improvements are calculated based on the performance of LightGCN. Note that we contrast the final layer with itself in the 1-layer XSimGCL. Based on Table 4, we have the following observations:

- In the vast majority of cases, the SGL variants, SimGCL and XSimGCL can largely outperform LightGCN. The

largest improvements are observed on Alibaba-iFashion where the performance of XSimGCL surpasses that of LightGCN by more than 25% under the 1-layer and 3-layer settings.

- SGL-ED and SGL-RW have very close performances and outperform SGL-ND by large margins. In many cases, SGL-WA show advantages over SGL-WA but still falls behind SGL-ED and SGL-RW. These results further corroborate that the InfoNCE loss is the primary factor which accounts for the performance gains, and meanwhile heuristic graph augmentations are not as effective as expected and some of them even degrade the performance.
- XSimGCL/SimGCL show the best/second best performance in almost all the cases, which demonstrates the effectiveness of the random noised-based data augmentation. Particularly, on the largest and sparsest dataset - Alibaba-iFashion, they significantly outperforms the SGL variants. In addition, it is obvious that the evolution from SimGCL to XSimGCL is successful, bringing non-negligible performance gains.

To further demonstrate XSimGCL's outstanding performance, we also compare it with a few recent augmentation-based or CL-based recommendation models. The implementations of these methods are available in our GitHub repository SELFRec as well. According to Table 6, XSimGCL and SimGCL still outperform with a great lead, achieving the best and the second best performance, respectively. NCL and MixGCF, which employ LightGCN as their backbones, also show their competitiveness. By contrast, DNN+SSL and BUIR are not as powerful as expected and even not comparable to LightGCN. We attribute their failure to: (1). DNNs are proved effective when abundant user/item features are provided. In our datasets, features are unavailable and the self-supervision signals are created by masking



TABLE 6: Performance comparison with other models.

Method	Yelp2018		Kindle		iFashion	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN	0.0639	0.0525	0.2057	0.1315	0.1053	0.0505
NCL	0.0670	0.0562	0.2090	0.1348	0.1088	0.0528
BUIR	0.0487	0.0404	0.0922	0.0528	0.0830	0.0384
DNN+SSL	0.0483	0.0382	0.1520	0.0989	0.0818	0.0375
MixGCF	0.0713	0.0589	0.2098	0.1355	0.1085	0.0520
SimGCL	0.0721	0.0601	0.2104	0.1374	0.1151	0.0567
XSimGCL	<b>0.0723</b>	<b>0.0604</b>	<b>0.2147</b>	<b>0.1415</b>	<b>0.1196</b>	<b>0.0586</b>

item embeddings, so it cannot fulfill itself in this situation. (2). In the paper of BUIR, the authors removed long-tail users and items to guarantee a good result, but we use all the data. We also notice that BUIR performs very well on suggesting popular items but poorly on long-tail items. This may explain why the original paper uses a biased experimental setting.

#### 4.2.2 Comparison of Training Efficiency

As has been claimed, XSimGCL is almost as lightweight as LightGCN in theory. In this part, we report the actual training time, which is more informative than the theoretical analysis. The reported figures are collected on a workstation with an Intel(R) Xeon(R) Gold 5122 CPU and a GeForce RTX 2080Ti GPU. These methods are implemented with Tensorflow 1.14, and a 2-layer setting is applied to all.

According to Fig. 6, we have the following observations:

- SGL-ED takes the longest time to finish the computation in a single batch, which is almost four times that of LightGCN on all the datasets. SimGCL ranks second due to its three-encoder architecture, which is almost two times that of LightGCN. Since SGL-WA, XSimGCL and LightGCN have the same architecture, their training costs for a batch are very close. The former two need a bit of extra time for the contrastive task.
- LightGCN is trained with hundreds of epochs, which is at least an order of magnitude more than the epochs that other methods need. By contrast, XSimGCL needs the fewest epochs to reach convergence and its predecessor SimGCL falls behind by several epochs. SGL-WA and SGL-ED require the same number of epochs to get converged and are slower than SimGCL. When it comes to the total training time, LightGCN is still the method trained with the longest time, followed by SGL-ED and SimGCL. Due to the simple architecture, SGL-WA and XSimGCL are the last two but XSimGCL only needs about half of the cost SGL-WA spends in total.

With these observations, we can easily draw some conclusions. First, CL can tremendously accelerate the training. Second, graph augmentations cannot contribute to the training efficiency. Third, the cross-layer contrasts not only brings performance improvement but also leads to faster convergence. By analyzing the gradients from the CL loss, we find that the noises in XSimGCL and SimGCL will add a small increment to the gradients, which works like a momentum and can explain the speedup. Compared with the final-layer contrast, the cross-layer has shorter route for gradient propagation. This can explain why XSimGCL needs fewer epochs compared with SimGCL.

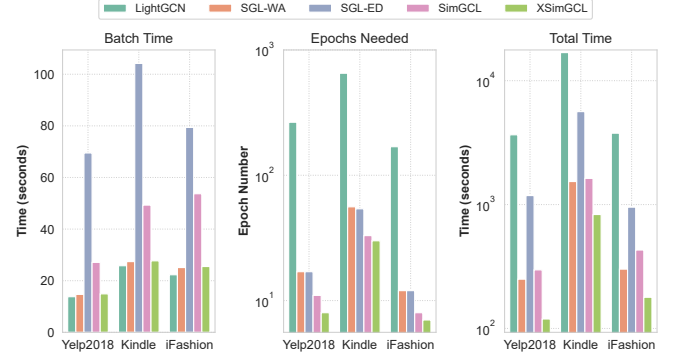


Fig. 6: The training speed of compared methods.

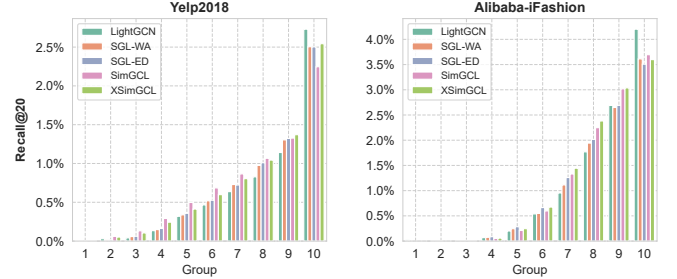


Fig. 7: The ability to promote long-tail items.

#### 4.2.3 Comparison of Ability to Promote Long-tail Items

Optimizing the InfoNCE loss is found to learn more evenly distributed representations, which is supposed to alleviate the popularity bias. To verify that XSimGCL upgrades this ability with the noise-based augmentation, we follow [12] to divide the test set into ten groups, labeled with IDs from 1 to 10. Each group includes the same number of interactions. The larger ID the group has, the more popular items it contains. We then conduct experiments with a 2-layer setting to check the Recall@20 value that each group achieves. The results are illustrated in Fig. 7.

According to Fig. 7, LightGCN is inclined to recommend popular items and achieves the highest recall value on the last group. By contrast, XSimGCL and SimGCL do not show outstanding performance on group 10, but they have distinct advantages over LightGCN on other groups. Particularly, SimGCL is the standout on Yelp2018 and XSimGCL keeps strong on iFashion. Their extraordinary performance in recommending long-tail items largely compensates for their loss on the popular item group. As for the SGL variants, they fall between LightGCN and SimGCL on exploring long-tail items and exhibit similar recommendation performance on Yelp2018. SGL-ED shows a slight advantage over SGL-WA on iFashion. Combining Fig. 3 with Fig. 7, we can easily find that the ability to promote long-tail items seems to positively correlate with the uniformity of representations. Since a good recommender system should suggest items that are most pertinent to a particular user instead of recommending popular items that might have been known, SimGCL and XSimGCL significantly outperforms other methods in this regard.

### 4.3 Hyperparameter Investigation

XSimGCL has three important hyperparameters:  $\lambda$  - the coefficient of the contrastive task,  $\epsilon$  - the magnitude of added noises, and  $l^*$  - the layer to be contrasted. In this part, we investigate the model's sensitivity to these hyperparameters.

#### 4.3.1 Influence of $\lambda$ and $\epsilon$

We try different combinations of  $\lambda$  and  $\epsilon$  with the set [0.01, 0.05, 0.1, 0.2, 0.5, 1] for  $\lambda$  and [0, 0.01, 0.05, 0.1, 0.2, 0.5] for  $\epsilon$ . We fix  $l^*=1$  and conduct experiments with a 2-layer setting, but we found that the best values of these two hyperparameters are also applicable to other settings. As shown in Fig. 8, on all the datasets XSimGCL reaches its best performance when  $\epsilon$  is in [0.05, 0.1, 0.2]. Without the added noises ( $\epsilon=0$ ), we can see an obvious performance drop. When  $\epsilon$  is too large ( $\epsilon=0.05$ ) or too small ( $\epsilon=0.01$ ), the performance declines as well. The similar trend is also observed by changing the value of  $\lambda$ . The performance is at the peak when  $\lambda=0.2$  on Yelp2018,  $\lambda=0.2$  on Amazon-Kindle, and  $\lambda=0.05$  on Alibaba-iFashion. According to our experience, XSimGCL (SimGCL) is more sensitive to  $\lambda$ , and  $\epsilon=0.1$  is usually a good and safe choice on most datasets. Besides, we also find that a larger  $\epsilon$  leads to faster convergence. But when it is overlarge (e.g., greater than 1), it will act like a large learning rate, causing the progressive zigzag optimization which will overshoot the minimum.

#### 4.3.2 Layer Selection for Contrast

In XSimGCL, two layers are chosen to be contrasted. We report the results of different choices in Fig. 9 where a 3-layer setting is used. Since these matrix-like heat maps are symmetric, we only display the lower triangular parts. The figures in the diagonal cells represent the results of contrasting the same layer. As can be seen, although the best choice varies from dataset to dataset, it always appears as the contrast between the final layer and one of the previous layers. We analyzed the similarities between representations of different layers and tried to find if  $l^*$  is related to the similarity but no evidence was found. Fortunately, XSimGCL usually achieves the best performance with a 3-layer setting, which means three attempts are enough. The amount of manual work for tuning  $l^*$  is therefore greatly reduced. A compromised way without tuning  $l^*$  is to randomly choose a layer in every mini-batch and contrast its embeddings with the final embeddings. We report the results of this random selection in the upper right of the heatmap. They are acceptable but much lower than the best performance.

### 4.4 Applicability Investigation

The noise-based CL has been proved effective when combining with LightGCN. We further wonder whether this method is applicable to other common backbones such as MF and GCN. Besides, whether uniform noises are the best choice remains unknown. In this part, we investigate the applicability of the noise-based augmentation and different types of noises.

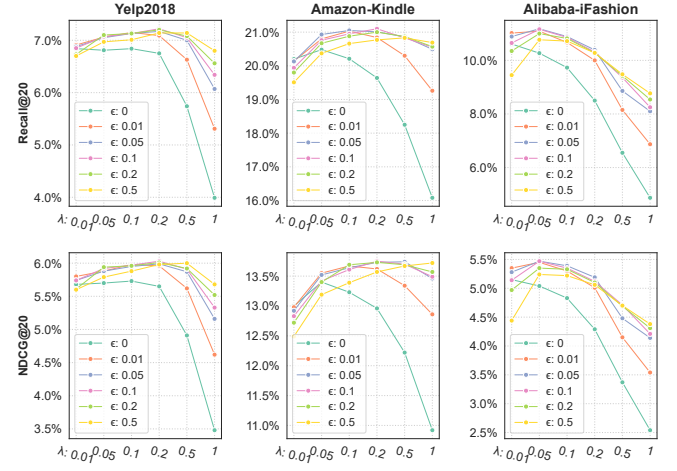


Fig. 8: The influence of  $\lambda$  and  $\epsilon$ .



Fig. 9: The influence of the layer selection for contrast.

#### 4.4.1 Noised-Based CL on Other Structures

We choose MF and the vanilla GCN as the backbones to be tested because these two simple backbones are widely used in practice. For MF that cannot adopt the cross-layer contrast, we add different uniform noises to the input embeddings for different augmentations. We tried many combinations of  $\lambda$  and  $\epsilon$  on these two structures and report the best results in Table 7 where NBC is short for noise-based CL. As can be seen, NBC can likewise improve GCN. We guess this is because GCN also has an aggregation mechanism. However, NBC cannot consistently improve MF. On the dataset of Amazon-Kindle it works, whereas on the other two datasets, it lowers the performance. This inconsistent effect cannot be easily concluded, and we leave it to our future work.

TABLE 7: Performance comparison of different backbones.

Method	Yelp2018		Kindle		iFashion	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF	0.0543	0.0445	0.1751	0.1068	0.996	0.468
MF + NBC	0.0517	0.0433	0.1878	0.1175	0.975	0.0453
GCN	0.0556	0.0452	0.1833	0.1137	0.952	0.0458
GCN + NBC	0.0632	0.0530	0.1989	0.1290	0.1017	0.0486

#### 4.4.2 XSimGCL with Other Types of Noises

We test three other types of noises in this experiment including adversarial perturbation obtained by following FGSM [36] (denoted by XSimGCL<sub>a</sub>), positive uniform noises without the sign of learned embeddings (denoted by XSimGCL<sub>p</sub>), and Gaussian noises (denoted by XSimGCL<sub>g</sub>). We tried many combinations of  $\lambda$  and  $\epsilon$  for different types of noises and presented the results in Table 8. As observed, the vanilla XSimGCL with signed uniform noises outperforms other variants. Although positive uniform noises and Gaussian noises also bring hefty performance gains compared with LightGCN, adding adversarial noises unexpectedly leads to a large drop of performance. This indicates that only a few particular distributions can generate helpful noises. Besides, the result that XSimGCL outperforms XSimGCL<sub>p</sub> demonstrates the necessity of the sign constraint.

TABLE 8: Performance comparison of different XSimGCL variants.

Method	Yelp2018		Kindle		iFashion	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN	0.0639	0.0525	0.2057	0.1315	0.1053	0.0505
XSimGCL <sub>a</sub>	0.0558	0.0464	0.1267	0.0833	0.0158	0.0065
XSimGCL <sub>p</sub>	0.0714	0.0596	0.2121	0.1398	0.1183	0.0577
XSimGCL <sub>g</sub>	<u>0.0722</u>	<u>0.0602</u>	<u>0.2140</u>	<u>0.1410</u>	<u>0.1190</u>	<u>0.0583</u>
XSimGCL	<b>0.0723</b>	<b>0.0604</b>	<b>0.2147</b>	<b>0.1415</b>	<b>0.1196</b>	<b>0.0586</b>

## 5 RELATED WORK

### 5.1 GNNs-Based Recommendation Models

In recent years, graph neural networks (GNNs) [41], [42] have brought the regime of DNNs to an end, and become a routine in recommender systems for its extraordinary ability to model the user behavior data. A great number of recommendation models developed from GNNs have achieved greater than ever performances in different recommendation scenarios [40], [13], [28], [43], [44]. Among numerous variants of GNNs, GCN [45] is the most prevalent one and drives many state-of-the-art graph neural recommendation models such as NGCF [46], LightGCN [28], LR-GCCF [47] and LCF [48]. Despite varying implementation details, all these GCN-based models share a common scheme which is to aggregate information from the neighborhood in the user-item graph layer by layer [42]. Benefitting from its simple structure, LightGCN becomes one of the most popular GCN-based recommendation models. It follows SGC [49] to remove the redundant operations in the vanilla GCN including transformation matrices and nonlinear activation functions. This design is proved efficient and effective for recommendation where only the user-item interactions are provided. It also inspires a lot of CL-based recommendation models such as SGL [12], NCL [18] and SimGCL [24].

### 5.2 Contrastive Learning for Recommendation

Contrastive learning [1], [2] recently has drawn considerable attention in many fields due to its ability to deal with massive unlabeled data [6], [5], [4]. As CL usually works in a self-supervised manner [3], it is inherently a sliver bullet to the data sparsity issue [50] in recommender systems.

Inspired by the success of CL in other fields, the community also has started to integrate CL into recommendation [15], [12], [14], [17], [13], [51], [52], [16]. The fundamental idea behind existing contrastive recommendation methods is to regard every instance (e.g., user or item) as a class, and then pull views of the same instance closer and push views of different instances apart when learning representations, where the views are augmented by applying different transformations to the original data. In brief, the goal is to maximize the mutual information between views from the same instance. Through this process, the recommendation model is expected to learn the essential information of the user-item interactions.

To the best of our knowledge, S<sup>3</sup>-Rec [15] is the first work that combines CL with sequential recommendation. It first randomly masks part of attributes and items to create sequence augmentations, and then pre-trains the Transformer [53] by encouraging the consistency between different augmentations. The similar idea is also found in a concurrent work CL4SRec [25], where more augmentation approaches including item reordering and cropping are used. Besides, S<sup>2</sup>-DHCN [14] and ICL [18] adopt advanced augmentation strategies by re-organizing/clustering the sequential data for more effective self-supervised signals. Qiu *et al.* proposed DuoRec [54] which adopts a model-level augmentation by conducting dropout on the encoder. Xia *et al.* [55] integrated CL into a self-supervised knowledge distillation framework to transfer more knowledge from the server-side large recommendation model to resource-constrained on-device models to enhance next-item recommendation. In the same period, CL was also introduced to different graph-based recommendation scenarios. S<sup>2</sup>-MHGN [27] and SMIN [56] integrate CL into social recommendation. HHGR [26] proposes a double-scale augmentation approach for group recommendation and develops a finer-grained contrastive objective for users and groups. CCDR [57] and CrossCBR [58] have explored the use of CL in cross-domain and bundle recommendation. Yao *et al.* [27] proposed a feature dropout-based two-tower architecture for large-scale item recommendation. NCL [18] designs a prototypical contrastive objective to capture the correlations between a user/item and its context. SEPT [17] and COTREC [59] further propose to mine multiple positive samples with semi-supervised learning on the perturbed graph for social/session-based recommendation. The most widely used model is SGL [12] which replies edge/node dropout to augment the graph data. Although these methods have demonstrated their effectiveness, they pay little attention to why CL can enhance recommendation.

## 6 CONCLUSION

In this paper, we revisit the graph CL in recommendation and investigate how it enhances graph recommendation models. The findings are surprising that the InfoNCE loss is the decisive factor which accounts for most of the performance gains, whilst the elaborate graph augmentations only play a secondary role. Optimizing the InfoNCE loss leads to a more even representation distribution, which helps to promote the long-tail items in the scenario of recommendation. In light of this, we propose a simple yet

effective noise-based augmentation approach, which can smoothly adjust the uniformity of the representation distribution through CL. An extremely simple model XSimGCL is also put forward, which brings an ultralight architecture for CL-based recommendation. The extensive experiments on three large and highly sparse datasets demonstrate that XSimGCL is an ideal alternative of its graph augmentation-based counterparts.

## REFERENCES

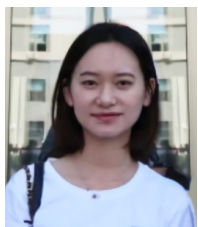
- [1] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2021.
- [2] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *arXiv preprint arXiv:2006.08218*, vol. 1, no. 2, 2020.
- [3] J. Yu, H. Yin, X. Xia, T. Chen, J. Li, and Z. Huang, "Self-supervised learning for recommender systems: A survey," *arXiv preprint arXiv:2006.07733*, 2022.
- [4] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *NeurIPS*, vol. 33, 2020.
- [5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020, pp. 1597–1607.
- [6] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," in *EMNLP*, 2021, pp. 6894–6910.
- [7] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020, pp. 9729–9738.
- [8] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar et al., "Bootstrap your own latent: A new approach to self-supervised learning," *NeurIPS*, 2020.
- [9] M. Singh, "Scalability and sparsity issues in recommender datasets: a survey," *Knowledge and Information Systems*, vol. 62, no. 1, pp. 1–43, 2020.
- [10] T. T. Nguyen, M. Weidlich, D. C. Thang, H. Yin, and N. Q. V. Hung, "Retaining data from streams of social platforms with minimal regret," in *IJCAI*, 2017, pp. 2850–2856.
- [11] T. Chen, H. Yin, Q. V. H. Nguyen, W.-C. Peng, X. Li, and X. Zhou, "Sequence-aware factorization machines for temporal predictive analytics," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1405–1416.
- [12] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *SIGIR*, 2021, pp. 726–735.
- [13] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang, "Self-supervised multi-channel hypergraph convolutional network for social recommendation," in *WWW*, 2021, pp. 413–424.
- [14] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," in *AAAI*, 2021, pp. 4503–4511.
- [15] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen, "S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization," in *CIKM*, 2020, pp. 1893–1902.
- [16] C. Zhou, J. Ma, J. Zhang, J. Zhou, and H. Yang, "Contrastive learning for debiased candidate generation in large-scale recommender systems," in *KDD*, 2021, pp. 3985–3995.
- [17] J. Yu, H. Yin, M. Gao, X. Xia, X. Zhang, and N. Q. V. Hung, "Socially-aware self-supervised tri-training for recommendation," in *KDD*, F. Zhu, B. C. Ooi, and C. Miao, Eds. ACM, 2021, pp. 2084–2092.
- [18] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," in *WWW*, 2022, pp. 2320–2329.
- [19] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," *NeurIPS*, pp. 15 509–15 519, 2019.
- [20] X. Zhou, A. Sun, Y. Liu, J. Zhang, and C. Miao, "Selfcf: A simple framework for self-supervised collaborative filtering," *arXiv preprint arXiv:2107.03019*, 2021.
- [21] D. Lee, S. Kang, H. Ju, C. Park, and H. Yu, "Bootstrapping user and item representations for one-class collaborative filtering," in *SIGIR*, F. Diaz, C. Shah, T. Suel, P. Castells, R. Jones, and T. Sakai, Eds., 2021, pp. 1513–1522.
- [22] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
- [23] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and debias in recommender system: A survey and future directions," *arXiv preprint arXiv:2010.03240*, 2020.
- [24] 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," in *SIGIR*, 2022, pp. 1294–1303.
- [25] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, J. Zhang, B. Ding, and B. Cui, "Contrastive learning for sequential recommendation," in *ICDE*. IEEE, 2022, pp. 1259–1273.
- [26] J. Zhang, M. Gao, J. Yu, L. Guo, J. Li, and H. Yin, "Double-scale self-supervised hypergraph learning for group recommendation," in *CIKM*, 2021, pp. 2557–2567.
- [27] T. Yao, X. Yi, D. Z. Cheng, F. Yu, T. Chen, A. Menon, L. Hong, E. H. Chi, S. Tjoa, J. Kang et al., "Self-supervised learning for large-scale item recommendations," in *CIKM*, 2021, pp. 4321–4330.
- [28] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *SIGIR*. ACM, 2020, pp. 639–648.
- [29] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*. AUAI Press, 2009, pp. 452–461.
- [30] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *WWW*, 2016, pp. 507–517.
- [31] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *ICML*, 2020, pp. 9929–9939.
- [32] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [33] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, "Kernel density estimation via diffusion," *The annals of Statistics*, vol. 38, no. 5, pp. 2916–2957, 2010.
- [34] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen, "Challenging the long tail recommendation," *Proc. VLDB Endow.*, vol. 5, no. 9, pp. 896–907, 2012.
- [35] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *AAAI*, vol. 34, no. 04, 2020, pp. 3438–3445.
- [36] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.
- [37] X. Zhang, F. X. Yu, S. Kumar, and S.-F. Chang, "Learning spread-out local feature descriptors," in *CVPR*, 2017, pp. 4595–4603.
- [38] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" *NeurIPS*, vol. 33, pp. 6827–6839, 2020.
- [39] T. Huang, Y. Dong, M. Ding, Z. Yang, W. Feng, X. Wang, and J. Tang, "Mixgcf: An improved training method for graph neural network-based recommender systems," pp. 665–674, 2021.
- [40] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *SIGIR*, 2020, pp. 1001–1010.
- [41] C. Gao, X. Wang, X. He, and Y. Li, "Graph neural networks for recommender system," in *WSDM*, 2022, pp. 1623–1625.
- [42] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *CSUR*, 2020.
- [43] J. Yu, H. Yin, J. Li, M. Gao, Z. Huang, and L. Cui, "Enhance social recommendation with adversarial graph convolutional networks," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
- [44] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *AAAI*, vol. 33, no. 01, 2019, pp. 346–353.
- [45] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
- [46] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *SIGIR*, 2019, pp. 165–174.
- [47] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *AAAI*, vol. 34, no. 01, 2020, pp. 27–34.



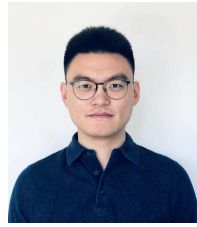
- [48] W. Yu and Z. Qin, "Graph convolutional network for recommendation with low-pass collaborative filters," in *ICML*, 2020, pp. 10936–10945.
- [49] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *ICML*, 2019, pp. 6861–6871.
- [50] J. Yu, M. Gao, J. Li, H. Yin, and H. Liu, "Adaptive implicit friends identification over heterogeneous network for social recommendation," in *CIKM*. ACM, 2018, pp. 357–366.
- [51] J. Ma, C. Zhou, H. Yang, P. Cui, X. Wang, and W. Zhu, "Disentangled self-supervision in sequential recommenders," in *KDD*, 2020, pp. 483–491.
- [52] Y. Wei, X. Wang, Q. Li, L. Nie, Y. Li, X. Li, and T.-S. Chua, "Contrastive learning for cold-start recommendation," in *ACM Multimedia*, 2021, pp. 5382–5390.
- [53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.
- [54] R. Qiu, Z. Huang, H. Yin, and Z. Wang, "Contrastive learning for representation degeneration problem in sequential recommendation," in *WSDM*, 2022, pp. 813–823.
- [55] X. Xia, H. Yin, J. Yu, Q. Wang, G. Xu, and Q. V. H. Nguyen, "On-device next-item recommendation with self-supervised knowledge distillation," in *SIGIR*, 2022, pp. 546–555.
- [56] X. Long, C. Huang, Y. Xu, H. Xu, P. Dai, L. Xia, and L. Bo, "Social recommendation with self-supervised metagraph informax network," in *CIKM*, 2021, pp. 1160–1169.
- [57] R. Xie, Q. Liu, L. Wang, S. Liu, B. Zhang, and L. Lin, "Contrastive cross-domain recommendation in matching," in *KDD*, 2022, pp. 4226–4236.
- [58] Y. Ma, Y. He, A. Zhang, X. Wang, and T.-S. Chua, "Crosscbr: Cross-view contrastive learning for bundle recommendation," *KDD*, pp. 1233–1241, 2022.
- [59] X. Xia, H. Yin, J. Yu, Y. Shao, and L. Cui, "Self-supervised graph co-training for session-based recommendation," in *CIKM*, 2021, pp. 2180–2190.



**Junliang Yu** received his B.S. and M.S. degrees in Software Engineering from Chongqing University, China. Currently, he is a final-year Ph.D. candidate at the School of Information Technology and Electrical Engineering, the University of Queensland. His research interests include recommender systems, social media analytics, and self-supervised learning.



**Xin Xia** received her B.S. degree in Software Engineering from Jilin University, China. Currently, she is a third-year Ph.D. candidate at the School of Information Technology and Electrical Engineering, the University of Queensland. Her research interests include knowledge distillation, sequence modeling, and self-supervised learning.



**Tong Chen** received his PhD degree in computer science from The University of Queensland in 2020. He is currently a Lecturer with the Data Science research group, School of Information Technology and Electrical Engineering, The University of Queensland. His research interests include data mining, recommender systems, user behavior modelling and predictive analytics.



**Lizhen Cui** is a full professor with Shandong University. He is appointed dean and deputy party secretary for School of Software, co-director of Joint SDU-NTU Centre for Artificial Intelligence Research(C-FAIR), director of the Research Center of Software and Data Engineering, Shandong University. His main interests include big data intelligence theory, data mining, wisdom science, and medical health big data AI applications.



**Nguyen Quoc Viet Hung** is a senior lecturer and an ARC DECRA Fellow in Griffith University. He earned his Master and PhD degrees from EPFL in 2010 and 2014 respectively. His research focuses on Data Integration, Data Quality, Information Retrieval, Trust Management, Recommender Systems, Machine Learning and Big Data Visualization, with special emphasis on web data, social data and sensor data.



**Hongzhi Yin** is an associate professor and ARC Future Fellow at the University of Queensland. He received his Ph.D. degree from Peking University, in 2014. His research interests include recommendation system, deep learning, social media mining, and federated learning. He is currently serving as Associate Editor/Guest Editor/Editorial Board for *ACM Transactions on Information Systems (TOIS)*, *ACM Transactions on Intelligent Systems and Technology (TIST)*, etc.