MixRec: Individual and Collective Mixing Empowers Data Augmentation for Recommender Systems

Anonymous Author(s)

Abstract

The core of the modern recommender systems lies in learning high-quality embedding representations of users and items to investigate their positional relations in the feature space. Unfortunately, data sparsity caused by difficult-to-access interaction data severely limits the effectiveness of recommender systems. Faced with such a dilemma, various types of self-supervised learning methods have been introduced into recommender systems in an attempt to alleviate the data sparsity through distribution modeling or data augmentation. However, most data augmentation relies on elaborate manual design, which is not only not universal, but the bloated and redundant augmentation process may significantly slow down model training progress. To tackle these limitations, we propose a novel Dual Mixing-based Recommendation Framework (MixRec) to empower data augmentation as we wish. Specifically, we propose individual mixing and collective mixing, respectively. The former aims to provide a new positive sample that is unique to the target (user or item) and to make the pair-wise recommendation loss benefit from it, while the latter aims to portray a new sample that contains group properties in a batch. The two mentioned mixing mechanisms allow for data augmentation with only one parameter that does not need to be set multiple times and can be done in linear time complexity. Besides, we propose the dual-mixing contrastive learning to maximize the utilization of these new-constructed samples to enhance the consistency between pairs of positive samples. Experimental results on four real-world datasets demonstrate the effectiveness of MixRec in terms of recommendation performance, training efficiency, sparsity resistance, and usability.

CCS Concepts

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

Keywords

recommender system, collaborative filtering, data augmentation, self-supervised learning

ACM Reference Format:

Anonymous Author(s). 2018. MixRec: Individual and Collective Mixing Empowers Data Augmentation for Recommender Systems. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As an integral part of modern Internet platforms, recommender systems have garnered significant attention [8, 21]. The wisdom of the crowd lies at the core of recommender systems, which leverage group behavior to filter out irrelevant information [8, 20]. With the rise of representation learning, the development of recommender systems has kept pace, with various neural networks being employed to model high-quality embeddings for users and items [13]. As a data-driven scientific application, recommender systems now rely on interaction data more than ever. However, for various reasons, it is often challenging for individual platforms to obtain comprehensive user data, leaving recommender systems frequently dealing with data sparsity [29, 41].

It is against this backdrop that many researchers have sought to introduce Self-supervised learning (SSL) [18] into recommender systems to mitigate the data sparsity problem. This contains two paradigms, Generative and Contrastive. Specifically, generative models [15, 41] based on probabilistic modeling can reconstruct historical interactions, while the more easily implemented contrastive learning [4] (CL) has made significant strides in combating data sparsity through data augmentation. Through data augmentation, contrastive learning maximizes mutual information from different views of the same sample, providing additional supervision signals for the recommendation task [23]. Consequently, research is increasingly focusing on exploring different types of data augmentation [36]. Early explorations concentrated on augmenting graph [34], where the graph is derived from user-item interactions [29]. Researcher constructs different views of the original input by randomly masking nodes or edges [2]. Subsequently, a new line of thought has emerged, focusing on achieving augmentation in the feature space. Examples include introducing noise for representations [32, 35], or finding semantic neighbors through clustering [16, 32], attention [19, 40], or hierarchical mechanisms [6].

Despite the impressive progress of these novel methods, we argue that existing research still faces several limitations. Firstly, much of CL-based methods tends to rely on manually crafted augmentation. Examples include the need to precisely control the ratio of censored raw data [29] or carefully add noise to raw data or representations to avoid over-corrupting the original semantic information [35]. More critically, to leverage more sophisticated augmentation strategies without overly disrupting recommendation, many recent methods introduce more hyper-parameters to balance multi-tasking [32, 35, 38]. Given the wide variability of data structures, researchers are often required to test with multiple hyper-parameter sets for different application scenarios. This not only increases the cost of trial and error, but also raises concerns about improving subsequent research. Although some methods adopt an alternative approach by achieving data augmentation through repeated sampling [9, 37], this strategy significantly increases warm-up and introduces unquantifiable sampling bias [3, 43].

Secondly, the data augmentation strategies proposed in many existing works substantially increase the time costs associated with model training and inference. In early research, commonly used data augmentation methods are often accompanied by multiple encodings and propagations to obtain different perspectives of the same input [29, 35]. And in more recent research, many studies introduce the paradigm of learnable (or adaptive) augmentation [19, 40], as well as unique feature-lookup mechanisms [16, 32] to identify suitable new views in the semantic space. In practice, these strategies typically lead to an exponential increase in model training and inference time, as multiple iterations of the base recommendation model's encoder are needed to construct representations of various views. As noted in previous pioneering work [29], the time complexity of the model after introducing graph augmentation-based CL is 3.7 times greater than that of the base model.

The final and most significant limitation is that many CL-based methods do not utilize these new samples efficiently. Specifically, the broad approach aims to enhance the consistency of a pair of positive samples while maintaining uniformity among the negative samples [4]. While effective, this approach is insufficient for fully utilizing each sample, particularly as it neglects to explore the varying characteristics of individual samples. If we intend to introduce a larger number of samples, we may encounter some challenges. In other words, the approach lacks scalability. In a nutshell, although data augmentation is essential for data-sparse recommendation scenarios, existing augmentation strategies often struggle with the dilemma. This prevents us from achieving desired augmentations, thereby limiting the overall recommendation performance. Therefore, we further investigate whether the following objectives can be achieved simultaneously:

- How can we more easily implement data augmentation with minimal hyper-parameter tuning?
- How can we achieve data augmentation without significantly increasing model complexity?
- How can new samples obtained through data augmentation be utilized more effectively to enhance recommendation?

To tackle the above limitations, we propose a novel Dual Mixing-based Recommendation Framework (**MixRec**) for recommender system. MixRec does not rely on any external strategies to construct new views of the original sample, instead using the samples themselves. Specifically, inspired by the mixing mechanism [39], we intend to create mixes of several samples. Based on this assumption, we propose the **Individual Mixing** and **Collective Mixing**, respectively. Individual mixing aims to create new views that are more closely aligned with the original sample while also incorporating information from other samples. Collective mixing, on the other hand, emphasizes a more holistic perspective, *i.e.*, the wisdom of the crowd. Such mixing process is a convex combination, which can be achieved with linear complexity and requires only simple controls to create new samples. In our experiments, we also find that the mixing process is hyper-parameter-agnostic and exhibits strong generalization properties. In addition, we propose the **Dual Mixing Contrastive Learning**, which, for the first time, maximizes the consistency of positive samples from dual mixing perspectives. This approach fully leverages all available new samples to enhance the supervision signals for the recommendation task. The major contributions of this paper are summarized as follows:

- We propose a recommendation framework MixRec, which contains two lightweight data augmentation strategies, individual mixing and collective mixing for recommendation task.
- We further propose dual-mixing contrastive learning, which maximizes the consistency of positive sample pairs from two perspectives while utilizing more negative samples to provide additional supervision signals for the recommendation task.
- We conduct comparison experiments with twenty related baseline methods across four real datasets to thoroughly validate the effectiveness of the proposed MixRec.

2 METHODOLOGY

2.1 Problem Formulation

Without loss of generality, a recommendation scenario contains M users ($\mathcal{U} = \{u_1, u_2, ..., u_M\}$) and N items ($\mathcal{U} = \{i_1, i_2, ..., i_N\}$), and interactions between them [8]. Based on existing work [7, 26], historical interactions are typically stored as an interaction matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$, where if there is an observed interaction between user u and item i, we then have $R_{ui} = 1$. The task of the recommender system aims to learn the prediction function $f_\Theta(u, j)$ as a means of predicting user u's preference score $\hat{R}_{uj}$ for the item j that have not been interacted with, where Θ is the set of trainable model parameter. For general recommendation scenarios, the only trainable parameters are the initial embedding representations for users $\{\mathbf{e}_u^{(0)}\}$ and items $\{\mathbf{e}_i^{(0)}\}$. Based on the above definition, we propose the dual mixing-based recommendation framework MixRec, as shown in Fig. 1.

2.2 Interaction Graph Encoding

In the context of a recommender system, the user-item interaction graph [26] is a natural way to represent the complex relationships between users and items. Specifically, the user-item interaction graph is denoted as $\mathcal{G} =< \mathcal{U} \cup \mathcal{I}, \mathcal{E} >$, where $\mathcal{U}$, $\mathcal{I}$, and $\mathcal{E}$ are the set of users, the set of items, and the interactions between users and items ($\mathcal{E}_{ui} = \mathcal{E}_{iu} = R_{ui}$). In recent studies, graph convolutional networks (GCNs) [12] are widely used to model interaction graph, which aggregate information from a node's neighbors, allowing each node to incorporate contextual information and leads to more informative representations. Given any user node $u \in \mathcal{U}$, the user embedding update process for an arbitrary layer is shown below:

$$\mathbf{e}_u^{(l)} = \mathrm{AGG}(\mathbf{e}_u^{(l-1)}, \{\mathbf{e}_i^{(l-1)} : i \in \mathcal{N}_u\}), \tag{1}$$

where $\mathbf{e}_u^{(l)} \in \mathbb{R}^d$ is the embedding of user u on the l-th layer, d is the embedding size, $\mathcal{N}_u$ is the first-order neighbor set of user u, and $\mathrm{AGG}(\cdot)$ is a manually defined aggregation function. The item side has a similar definition. In general, AGG functions have multiple implementation strategies. Considering that general recommendation only uses user and item ID as inputs, we adopt the widely used lightweight graph convolution [7] to encode user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ on the interaction graph:

$$\mathbf{e}_u^{(l)} = \sum_{i \in \mathcal{N}_u} p_{ui} \mathbf{e}_i^{(l-1)}; \quad \mathbf{e}_i^{(l)} = \sum_{u \in \mathcal{N}_i} p_{ui} \mathbf{e}_u^{(l-1)}, \tag{2}$$

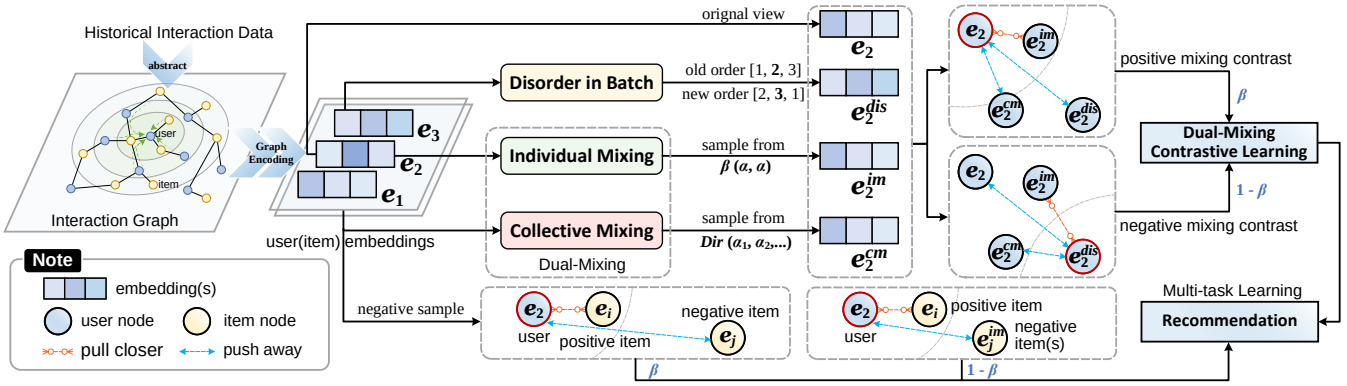


Figure 1: The complete information flow of the proposed MixRec. MixRec contains several phases of graph encoding, dual-mixing, dual-mixing contrastive learning, and multi-task learning for recommendation.

where $p_{ui} = 1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$ is the graph Laplacian norm [12, 26]. Compared to vanilla GCN [12], Eq. 2 does not rely on feature transformation, and the process of information aggregation is done only by linear combination of neighbors, which is more in line with the semantic-free feature of implicit feedback [8, 20]. After L layers of propagation, we construct usable embeddings for downstream tasks by addition [40]: $\mathbf{e}_u = \sum_{l=1}^{L} \mathbf{e}_u^{(l)}$ and $\mathbf{e}_i = \sum_{l=1}^{L} \mathbf{e}_i^{(l)}$.

2.3 Dual-Mixing for Data Augmentation

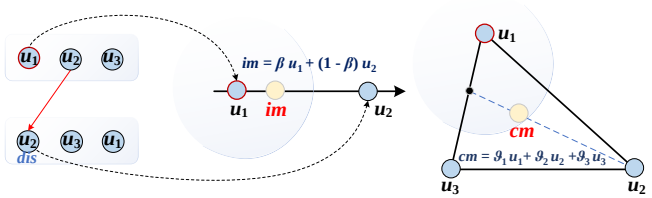
Although user and item representations obtained by graph structure encoding can already be directly used in the downstream recommendation task, a serious problem is the data sparsity issue as mentioned in the Introduction section [29, 41]. Extremely sparse implicit feedback may not provide sufficient supervision signals, leading to suboptimal learned embeddings [20]. Therefore, data augmentation is especially important for recommendation task. Given that most existing data augmentation strategies for self-supervised contrastive learning suffer from high complexity and a lack of flexibility, this dilemma prompts us to reconsider and rediscover a proven augmentation strategy. Our goal is to generate multiple new views as we wish, without relying on complex augmentation rules or time-consuming repetitive sampling. Building on this, we propose the **Dual-Mixing** for data augmentation, which incorporates both individual and collective mixing.

2.3.1 ***Individual Mixing.*** We first consider any individual (user u or item i) in a sampled mini-batch. By graph encoding, we have obtained their corresponding embedding representations: $\mathbf{e}_u$ and $\mathbf{e}_i$. Inspired by [10, 39], we construct new samples by linearly combining the original embedding pairs. Specifically, **individual mixing** creates synthetic training samples by linearly interpolating between two random samples from a sampled mini-batch:

$$\mathbf{e}_u^{im} = \beta_u \cdot \mathbf{e}_u + (1-\beta_u) \cdot \mathbf{e}_u^{dis}; \quad \mathbf{e}_i^{im} = \beta_i \cdot \mathbf{e}_i + (1-\beta_i) \cdot \mathbf{e}_i^{dis}, \tag{3}$$

where β is a mixing coefficient drawn from a symmetric Beta distribution $Beta(\alpha, \alpha)$ with shape parameter $\alpha \in (0, \infty)$ (For simplicity, we define $\alpha = \alpha_u = \alpha_i$), and $\mathbf{e}_u^{dis}$ is the embedding at the corresponding position of user u after the order of user embeddings in a batch has been disrupted (*i.e.*, the embedding of any other user within the same batch).

Unlike traditional data augmentation, which is commonly used in contrastive learning [34, 36], individual mixing does not require multiple instances of graph encoding or repeated sampling. More importantly, we can generate new views on demand by simply tweaking the shape parameter α, while embedding $\mathbf{e}_u^{dis}$ is automatically obtained through a disordered operation. In practice, we empirically set α to 0.1, which eliminates the need for extensive manual effort in individual mixing. In addition, by controlling for a smaller α, we constrain the sampled beta β to yield larger values, enabling the newly generated sample $\mathbf{e}_u^{im}$ to retain as many properties of the original sample $\mathbf{e}_u$ as possible. This aligns with the invariance of contrastive learning [14], treating these new sample as positive of the original view.



(a) disorder (b) individual mixing (c) collective mixing

Figure 2: Example of construction process for three new views of user node u_1 (batch size $|\mathcal{B}| = 3$).

2.3.2 ***Collective Mixing.*** With individual mixing, we can easily generate new views for user u or item i. These new views retain information from most of the original views, allowing us to align them in feature space by maximizing mutual information. However, we note that the essence of recommender systems lies in the wisdom of the crowd [7, 8], highlighting the importance of considering inter-user relationships. Therefore, we further propose **collective mixing**, building upon individual mixing, to create new views that incorporate group information for each user. Specifically, collective mixing generates new examples by forming convex combinations of pairs of examples from the entire batch:

$$\mathbf{e}_u^{cm} = \vartheta_1 \mathbf{e}_1 + \vartheta_2 \mathbf{e}_2 + \cdots + \vartheta_{|\mathcal{B}|} \mathbf{e}_{|\mathcal{B}|}, \quad \text{s.t.} \sum_{o=1}^{|\mathcal{B}|} \vartheta_o = 1.0, \tag{4}$$

Table 1: Notation and explanation of the original and other new views of the user u.

Notation	Explanation
$\mathbf{e}_u$	The original view of user u
$\mathbf{e}_u^{im}$	The new view of user u by individual mixing
$\mathbf{e}_u^{cm}$	The new view of user u by collective mixing
$\mathbf{e}_u^{dis}$	The new view of user u by disorder in a batch

where $\mathbf{e}_u^{cm}$ is the new view for user u generated through collective mixing, with a similar definition for the item side. $\{\vartheta_1, \vartheta_2, ..., \vartheta_{|\mathcal{B}|}\}$ is a set of coefficients for convex combination, which is sampled from a multivariate Dirichlet distribution [1]:

$$\{\vartheta_1, \vartheta_2, ..., \vartheta_{|\mathcal{B}|}\} \sim Dirichlet(\alpha_1, \alpha_2, ..., \alpha_{|\mathcal{B}|}), \tag{5}$$

where $\{\alpha_i : i \in \mathcal{B}\}$ is a set of positive real numbers to parameterize, and $\mathcal{B}$ is a sampled mini batch. The new sample $\mathbf{e}_u^{cm}$ contains more information from other users in the same batch than the individual mixing. The new samples obtained from collective mixing contain more information from other users than individual mixing.

We now sort out the proposed dual-mixing, and the process of constructing new views is presented in Fig. 2. For individual mixing (Fig. 2(b)), the new sample *im* is a linear interpolation of the original inputs u_1 and u_2. For collective mixing (Fig. 2(c)), the new sample *cm* produced by the convex combination always lies within the convex hull of the original inputs $\{u_1, u_2, u_3\}$. Due to the influence of multiple sets of shape arameters, the *cm* is affected by the influence from a wider range of users, which makes it extremely limited although it contains information from the original sample u_1. Therefore, the sample *im* obtained from individual mixing we regard as a **positive sample** of the original view u_1, while the sample *cm* produced by collective mixing is a **hard negative sample** of the original view u_1. And the sample *dis* (*i.e.,* u_2) obtained by disorder is naturally a **easy negative sample** of the original view u_1.

2.4 Dual-Mixing Contrastive Learning

Taking the user side as an example, through the proposed dual-mixing, we obtain two new views of user u (*i.e.,* $\mathbf{e}_u^{im}$ and $\mathbf{e}_u^{cm}$), , along with one additional view $\mathbf{e}_u^{dis}$ of the other users derived from disorder. For user u, we have a total of four different views, as outlined in Table 1. It is now time to consider how to utilize these new views to provide additional supervision signals for the main recommendation task. A reasonable approach is to apply widely-used infoNCE [4, 36] to maximize the mutual information between positive samples:

$$\mathcal{L}_u^{\text{cl}} = \frac{\exp(s(\mathbf{e}'_u, \mathbf{e}''_u)/\tau)}{\sum_{v\in\mathcal{U}}\exp(s(\mathbf{e}'_u, \mathbf{e}''_v)/\tau)}, \tag{6}$$

where $\mathbf{e}'_u$ and $\mathbf{e}''_u$ are additional views of user u via data augmentation, $s(\cdot,\cdot)$ is the cosine similarity function, and τ is a temperature parameter. The contrastive loss essentially encourages the alignment of positive views while pushing the positive view $\mathbf{e}'_u$ away from the negative sample view $\mathbf{e}''_v$, thereby making the feature space more uniform [23, 24]. However, we argue that the supervision signals provided by vanilla contrastive learning are insufficient, as it fails to account for the uniformity across a wider range of sample pairs. In addition, the standard infoNCE loss overlooks the original view, leading to inconsistency between the auxiliary task and the main recommendation task. Therefore, we shift our focus and leverage the multiple views constructed earlier to propose the **dual-mixing contrastive learning**. Given user u, we first define the positive mixing contrastive loss:

$$\mathcal{L}_u^{\text{pos}} = -\log\frac{\exp(s(\mathbf{e}_u, \mathbf{e}_u^{im})/\tau)}{\sum_{v\in\mathcal{U}}[\exp(s(\mathbf{e}_u, \mathbf{e}_v^{dis})/\tau) + \exp(s(\mathbf{e}_u, \mathbf{e}_v^{cm})/\tau)]}. \tag{7}$$

In contrast to the original infoNCE, the above equation directly utilizes the original view as an anchor node to align the main task with the auxiliary task. In addition, we expand the number of negative sample pairs by contrasting the original view with multiple negative samples, further optimizing the uniformity of the entire feature space. To fully leverage the efficacy of these views, we next present the negative mixing contrastive loss as a counterpart:

$$\mathcal{L}_u^{\text{neg}} = -\log\frac{\exp(s(\mathbf{e}_u^{dis}, \mathbf{e}_u^{im})/\tau)}{\sum_{v\in\mathcal{U}}[\exp(s(\mathbf{e}_u^{dis}, \mathbf{e}_v)/\tau) + \exp(s(\mathbf{e}_u^{dis}, \mathbf{e}_v^{cm})/\tau)]}. \tag{8}$$

In this step, we boldly employ the negative view $\mathbf{e}_u^{cm}$ at the corresponding position, obtained through order perturbation, as the anchor node while maintaining the roles of the other views. This design aims to effectively measure the correlation between these views, thereby providing additional supervision signals for the recommendation task. Note that the positive sample $\mathbf{e}_u^{im}$ is derived from mixing the two anchor nodes $\mathbf{e}_u$ and $\mathbf{e}_u^{dis}$. Consequently, both contrastive losses include a measure of the alignment between $\mathbf{e}_u^{im}$ with the two anchor nodes. The similarity of the positive sample $\mathbf{e}_u^{im}$ to the two anchor nodes is governed by the mixing coefficient β_u. Therefore, we similarly utilize β_u to determine the weight of the two contrastive losses described above:

$$\mathcal{L}_{\text{user}} = \sum_{u\in\mathcal{U}} \beta_u \cdot \mathcal{L}_u^{\text{pos}} + (1-\beta_u)\cdot\mathcal{L}_u^{\text{neg}}, \tag{9}$$

where β_u is the mixing coefficient used for individual mixing in Eq. 3. For the item side, we have a similar definition:

$$\mathcal{L}_{\text{item}} = \sum_{i\in\mathcal{I}} \beta_i \cdot \mathcal{L}_i^{\text{pos}} + (1-\beta_i)\cdot\mathcal{L}_i^{\text{neg}}. \tag{10}$$

2.5 Multi-task Learning

The optimization of MixRec involves two components: the recommendation task and the auxiliary task. For the main recommendation task, we take the widely applied BPR loss [20] to make more variability between positive and negative samples:

$$\mathcal{L}_{\text{BPR}}^{\text{pos}} = \sum_{<u,i>\in O^+, <u,j>\in O^-} -\ln\sigma(\mathbf{e}_u^\top\mathbf{e}_i - \mathbf{e}_u^\top\mathbf{e}_j), \tag{11}$$

where σ is the sigmoid function. $i \in \mathcal{I}$ is an item that user $u \in \mathcal{U}$ has interacted with, and $j \in \mathcal{I}$ is any uninteracted one, and both of them are sampled from a uniform distribution [9]. O^+ and O^- are the observed and unobserved interaction sets, respectively.

As mentioned earlier, the original interaction data is highly sparse, making it infeasible to achieve satisfactory results by relying solely on the BPR loss described above. Therefore, we further utilize the previously mentioned individual mixing to construct additional negative samples for each user: $\mathbf{e}_j^{im} = \beta_i \cdot \mathbf{e}_j + (1-\beta_i)\cdot\mathbf{e}_j^{dis}$, where

item j is the original negative sample for user u, and $\mathbf{e}_j^{dis}$ is constructed by disorder. Thus, within a mini-batch, we can construct multiple new negative samples for user u, resulting in the set O_u^-. Subsequently, we compute the pairwise ranking loss for user u based on these new negative samples:

$$\mathcal{L}_{\text{BPR}}^{\text{neg}} = \sum_{<u,i>\in O^+} -\ln\sigma(\mathbf{e}_u^\top \mathbf{e}_i - \sum_{j\in O_u^-} \mathbf{e}_u^\top \mathbf{e}_j^{im}). \tag{12}$$

Compared to the original BPR loss, the equation above further considers the distance relationship between positive sample i and multiple constructed negative samples O_u^-, which encourages item i to remain closer to user u in the feature space. Similarly, we balance the two BPR losses through a linear combination:

$$\mathcal{L}_{\text{main}} = \beta_i \cdot \mathcal{L}_{\text{BPR}}^{\text{pos}} + (1-\beta_i) \cdot \mathcal{L}_{\text{BPR}}^{\text{neg}}. \tag{13}$$

Finally, to integrate the recommendation task with the auxiliary task, we employ a multi-task joint training strategy for optimization. The complete optimization objective of MixRec is defined as follows:

$$\mathcal{L}_{\text{MixRec}} = \mathcal{L}_{\text{main}} + \lambda_1 \cdot (\mathcal{L}_{\text{user}} + \mathcal{L}_{\text{item}}) + \lambda_2 \cdot \|\Theta\|_2^2, \tag{14}$$

where λ_1 and λ_2 are hyper-parameters to trade off the magnitude of losses, and $\Theta = \mathbf{E}^{(0)}$ is the set of trainable model parameter.

2.6 Time Complexity

In this section, we analyze the time complexity of MixRec. Specifically, we set the number of nodes and edges of the interaction graph $\mathcal{G}$ to be $|\mathcal{V}|$ and $|\mathcal{E}|$, respectively. L is the number of GCN layers, d is the embedding size, and $|\mathcal{B}|$ is the batch size. Next, we present the key components that contribute to the time complexity:

- **Interaction Graph Encoding:** The time complexity of this component is in line with mainstream methods since we adopt the design of the classical LightGCN [7]. Therefore, the time complexity of this component is $O(2|\mathcal{E}|Ld)$.
- **Dual-Mixing:** We introduce individual and collective mixing for data augmentation. Recalling Eqs. 3 and 4, this component does not significantly increase the time complexity, as the mixing operation involves only an addition of embeddings rather than matrix multiplication
- **Dual-Mixing Contrastive Learning:** In this component, we need to compute the contrastive loss on the user and item side, respectively. Therefore, the time complexity of this component is $O(4(|\mathcal{B}|d + 2|\mathcal{B}|^2 d))$.
- **Recommendation Losses:** We adopt the widely used BPR loss [20] to optimize the recommendation task. In addition, we additionally introduced mixed negative samples. Therefore, the time complexity of this component is $O(2|\mathcal{B}|d + |\mathcal{B}|^2 d)$.

In practice, the time complexity of MixRec comes mainly from interaction graph encoding since the batch size $|\mathcal{B}|$ is much smaller than the interaction scale $|\mathcal{E}|$. As a result, the actual complexity of MixRec is slightly higher than that of LightGCN due to the additional computation of losses; however, it remains significantly lower than other methods based on self-supervised contrastive learning or multiple sampling. This is primarily because MixRec requires only linear time complexity for data augmentation, avoiding the need to perform graph encoding multiple times. Additionally, MixRec does not require extra time for sampling multiple negative samples.

Table 2: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Sparsity
Douban-Book	13,024	22,347	792,062	99.72%
Yelp	31,668	38,048	1,561,406	99.87%
Tmall	47,939	41,390	2,357,450	99.88%
Amazon-Book	52,643	91,599	2,984,108	99.94%

3 Experiments

In this section, we perform experiments on four real-world datasets to validate our proposed MixRec compared with state-of-the-art recommendation methods.

3.1 Experimental Settings

3.1.1 ***Datasets.*** To validate the effectiveness of MixRec, we adopt four widely used recommendation datasets: Douban-Book [32, 35], Yelp [7, 35], Amazon-Book [7, 35], and Tmall [19, 40], which are varied in field, scale, and sparsity level. Detailed statistics for the four datasets are presented in Table 2. For fair comparison, preprocessing of all datasets remains consistent with previous studies [7, 29]. Specifically, all explicit feedback is forced to be converted to implicit feedback (binary values). Items that a user has interacted with are considered positive samples, while all other items are considered negative samples that can be sampled for that user.

3.1.2 ***Baselines.*** To validate the effectiveness of MixRec, we choose the following state-of-the-art recommendation methods for comparison experiment:

- **Factorization-based method**: MF [20].
- **Generative methods**: Mult-VAE [15], CVGA [41], and DiffRec [25].
- **GCN-based methods**: NGCF [26], LightGCN [7], IMP-GCN [17], MixGCF [9], and CAGCN* [28].
- **SSL-based methods**: SGL-ED [29], NCL [16], DirectAU [22], SimGCL [35], GraphAU [31], CGCL [6], VGCL [32], LightGCL [2], SCCF [30], RecDCL [38], and BIGCF [40].

3.1.3 ***Hyperparameter Settings.*** We implement MixRec in PyTorch[1]. For a fair comparison, we adopt an experimental setup consistent with previous works [7, 35]. Specifically, the embedding size and batch size of all models are set to 64 (excluding Mult-VAE [15], DiffRec [25], and RecDCL [38]) and 2048, respectively. For all graph-based methods, the number of network layers was set to 3 [7] (excluding IMP-GCN [17]). The default optimizer is Adam [11], and initialization is done via the Xavier method [5]. We follow the suggested settings in the authors' original papers and use a grid search to choose the optimum hyperparameters for all baselines. For MixRec, we empirically set the temperature coefficient τ to be 0.2. The weight of contrastive learning λ_1 is set in the range of {0.01, 0.5, 0.1, 0.2, ..., 2.0}, and the weight of L_2 regularization λ_2 is set in $1e^{-4}$ by default. The default setting for the shape parameter α is 0.1. To assess the performance of Top-N recommendation, we employ two commonly used evaluation metrics: Recall@N and NDCG@N (N=20 by default), which are computed by the all-ranking [7, 26, 35].

[1]https://anonymous.4open.science/r/MixRec-2207

Table 3: Overall performance comparisons on Yelp, Amazon-Book, Tmall, and Douban-Book datasets. The results of MixRec are bolded, whereas the best baseline is underlined. * denotes that the improvement is significant with a p-value < 0.001 based on a two-tailed paired t-test. Part of the results are duplicated from original papers for consistency.

Model Name	Year	Yelp		Amazon-Book		Tmall		Douban-Book	
		Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
MF [20]	UAI'09	0.0539	0.0439	0.0308	0.0239	0.0547	0.0400	0.1292	0.1147
Mult-VAE [15]	WWW'18	0.0584	0.0450	0.0407	0.0315	0.0740	0.0552	0.1670	0.1604
CVGA [41]	TOIS'23	0.0694	0.0571	0.0492	0.0379	0.0854	0.0648	0.1736	0.1650
DiffRec [25]	SIGIR'23	0.0665	0.0556	0.0514	0.0418	0.0792	0.0612	0.1619	0.1661
NGCF [26]	SIGIR'19	0.0560	0.0456	0.0342	0.0261	0.0629	0.0465	0.1376	0.1215
LightGCN [7]	SIGIR'20	0.0639	0.0525	0.0411	0.0315	0.0711	0.0530	0.1504	0.1404
MixGCF [9]	KDD'21	0.0713	0.0589	0.0485	0.0378	0.0813	0.0611	0.1731	0.1685
IMP-GCN [17]	WWW'21	0.0653	0.0531	0.0460	0.0357	0.0729	0.0539	0.1725	0.1604
CAGCN* [17]	WWW'23	0.0711	0.0590	0.0506	0.0400	0.0783	0.0581	0.1704	0.1667
SGL-ED [29]	SIGIR'21	0.0675	0.0555	0.0478	0.0379	0.0738	0.0556	0.1633	0.1585
NCL [16]	WWW'22	0.0685	0.0577	0.0481	0.0373	0.0750	0.0553	0.1647	0.1539
DirectAU [22]	KDD'22	0.0703	0.0583	0.0506	0.0406	0.0752	0.0576	0.1660	0.1568
SimGCL [35]	SIGIR'22	0.0721	0.0601	0.0515	0.0414	0.0884	0.0674	0.1728	0.1671
GraphAU [31]	CIKM'23	0.0691	0.0574	0.0508	0.0403	0.0840	0.0625	0.1699	0.1633
CGCL [6]	SIGIR'23	0.0694	0.0561	0.0482	0.0375	0.0861	0.0650	0.1741	0.1667
VGCL [32]	SIGIR'23	0.0715	0.0587	0.0506	0.0401	0.0880	0.0670	0.1733	0.1689
LightGCL [32]	ICLR'23	0.0692	0.0571	0.0506	0.0397	0.0833	0.0637	0.1570	0.1455
SCCF [30]	KDD'24	0.0701	0.0580	0.0491	0.0399	0.0772	0.0580	0.1711	0.1639
RecDCL [38]	WWW'24	0.0690	0.0560	0.0510	0.0405	0.0853	0.0632	0.1664	0.1526
BIGCF [40]	SIGIR'24	0.0729	0.0602	0.0500	0.0398	0.0876	0.0664	0.1741	0.1682
MixRec (Ours)		**0.0740***	**0.0612***	**0.0541***	**0.0433***	**0.0900***	**0.0686***	**0.1778***	**0.1712***
p-values		5.21e-6	3.97e-5	2.07e-7	9.14e-6	4.75e-5	1.25e-5	9.82e-6	4.44e-5

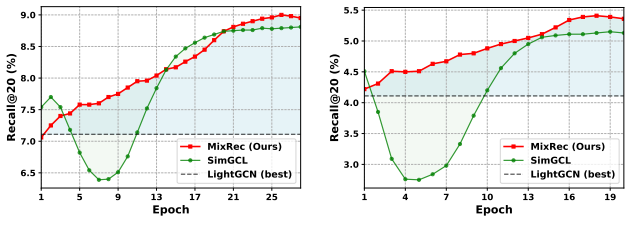
3.2 Performance Comparisons

3.2.1 ***Overall Comparisons.*** Table 3 shows the performance of MixRec and all baseline methods on four datasets. MixRec achieves the best recommendation performance over all baselines on all datasets. Quantitatively, MixRec improves over the best baselines *w.r.t.* Recall@20 by 1.78%, 5.05%, 1.81%, and 2.13% on Yelp, Amazon-Book, Tmall, and Douban-Book datasets, respectively. The experimental results demonstrate the effectiveness and generalization of MixRec. We attribute the performance improvement to the proposed individual and collective mixing, which effectively achieves data augmentation and alleviates the data sparsity problem faced by recommender systems.

MixGCF [9], another recommendation model utilizing the mixing mechanism, achieves better recommendation performance than MF and LightGCN, further demonstrating the superiority of the mixing mechanism. However, MixGCF merely samples multiple negative instances for calculating the BPR loss, which not only introduces sampling bias but also fails to provide additional supervision signals for the recommendation task. Compared to all contrastive learning-based methods, MixRec consistently outperforms them. This can be attributed to the fact that traditional contrastive learning methods do not fully leverage both positive and negative samples. In contrast, MixRec introduces dual-mixing contrastive learning, which effectively evaluates the role of various negative samples. Moreover, MixRec avoids the need for multiple graph encodings, ensuring its time complexity remains relatively low.

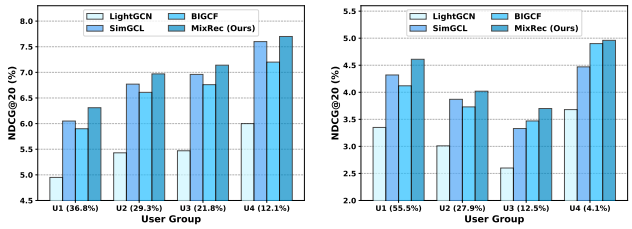
Table 4: Efficiency comparison on Tmall and Amazon-Book datasets *w.r.t.* time/epoch (T/E), number of epochs, and total runtime (measured in seconds (s), minutes (m), hours (h)).

	Tmall			Amazon-Book		
	T/E	epochs	runtime	T/E	epochs	runtime
LightGCN	49.1s	286	3h50m	54.7s	423	6h26m
IMP-GCN	224.5s	220	13h43m	357.2s	260	25h48m
MixGCF	180.3s	114	5h43m	202.5s	89	5h
SimGCL	132.4s	24	53m	167.5s	21	58m
CGCL	105.5s	76	2h14m	147.3s	59	2h25m
BIGCF	56.7s	40	38m	71.1s	42	50m
MixRec-1	32.4s	34	**18m**	35.3s	26	**15m**
MixRec-3	56.1s	26	**24m**	61.5s	19	**19m**



(a) Tmall (b) Amazon-Book

Figure 3: Training curves of LightGCN (best), SimGCL and MixRec on (a) Tmall and (b) Amazon-Book datasets.



(a) Tmall (b) Amazon-Book

Figure 4: Sparsity tests on (a) Tmall and (b) Amazon-Book datasets. The x-axis shows user groups and proportions.

3.2.2 ***Comparisons w.r.t. Efficiency.*** In this section, we present a comparison of the training time of MixRec ($L = 1$ and $L = 3$) with baseline methods on the two largest datasets, Tmall and Amazon-Book, as shown in Table 4. As can be seen, while MixRec-3's single training time is only slightly higher than LightGCN [7] due to the additional loss computations, it is still significantly lower than other methods. MixRec-1, on the other hand, takes even less time than LightGCN. Besides, methods like LightGCN are constrained by sparse interaction data, requiring hundreds of iterations to achieve convergence. In contrast, MixRec converges in far fewer iterations, resulting in a significantly reduced overall training time.

We further compare the training process of MixRec with the best-performing baseline method SimGCL [35] on Tmall and Amazon-Book datasets, as shown in Fig. 3. Compared to SimGCL, MixRec's training process is more stable, as evidenced by the absence of the performance drop seen in the early stages. We attribute this stability to MixRec's ability to strike a better balance between alignment and uniformity, preventing early excessive uniformity from disrupting the effective distribution of the feature space [24].

3.2.3 ***Comparisons w.r.t. Data Sparsity.*** In this section, we study the sparsity resistance of MixRec with the classical method LightGCN and the two well-performing baseline methods SimGCL and BIGCF on the two sparsest datasets, Tmall and Amazon-Book datasets. We take a generalized approach by dividing the interaction data from the training set evenly into four user groups $\{U_1, U_2, U_3, U_4\}$ based on the scale of the interactions. Specifically, $U1$ has the fewest interactions per user, indicating that it is the sparsest user group. And so on, $U4$ is the most active engaging user group. We train on the full training set and test each user group individually, and the experimental results are shown in Fig. 4.

MixRec achieves noticeable performance gains across all sparse groups, further demonstrating its effectiveness. Focusing on the sparsest group $U1$, the improvement rates on the two datasets are 12% and 10%, respectively. We attribute this performance improvement primarily to the proposed individual and collective mixing strategies. These not only significantly increase the number of samples but also provide additional supervision signals for main recommendation task through dual-mixing contrastive learning.

3.2.4 ***Comparisons w.r.t. GCN Layers.*** Table 5 provides comparisons of the effectiveness of MixRec and other methods with various GCN layer settings. It should be noted that MixRec with only one layer outperforms even SimGCL [35] and BIGCF [40] with three layers, which shows MixRec can effectively mine user-item relationships without over-reliance on high-order graph structures, making it effective in reducing training costs (MixRec-1 in Table 4).

Table 5: Performance comparisons between MixRec and other baseline methods *w.r.t.* number of GCN layer L.

# Layers	Method	Tmall		Amazon-Book	
		Recall	NDCG	Recall	NDCG
$L = 1$	SimGCL [35]	0.0834	0.0635	0.0453	0.0358
	BIGCF [40]	0.0851	0.0648	0.0466	0.0360
	MixRec	**0.0890**	**0.0681**	**0.0533**	**0.0429**
$L = 2$	SimGCL [35]	0.0867	0.0665	0.0507	0.0405
	BIGCF [40]	0.0865	0.0660	0.0493	0.0401
	MixRec	**0.0896**	**0.0684**	**0.0535**	**0.0431**
$L = 3$	SimGCL [35]	0.0884	0.0674	0.0515	0.0414
	BIGCF [40]	0.0876	0.0664	0.0500	0.0398
	MixRec	**0.0900**	**0.0686**	**0.0541**	**0.0433**

3.3 In-depth Studies of MixRec

3.3.1 ***Ablation Studies.*** We construct a series of variants to verify the validity of each module in MixRec:

- $\text{MixRec}_{\text{w/o DMCL (user)}}$: remove the Dual-Mixing Contrastive Learning on the user side (Eq. 9);
- $\text{MixRec}_{\text{w/o DMCL (item)}}$: remove the Dual-Mixing Contrastive Learning on the item side (Eq. 10);
- $\text{MixRec}_{\text{w/o IM}}$: remove individual mixing (Eq. 3), and modify the positive sample to be the anchor node itself;
- $\text{MixRec}_{\text{w/o DM}}$: remove collective mixing (Eq. 4).

The experimental results for all variants with MixRec on four datasets are shown in Table 6. It is obvious that removing any of the modules resulted in varying degrees of performance degradation for MixRec, demonstrating the effectiveness of the various modules. Regarding the DMCL modules on both the user and item sides, the most significant performance degradation occurs when these modules are removed, indicating that relying solely on the main recommendation task is insufficient for modeling high-quality user and item embeddings. Focusing on the other three modules that utilize data augmentation, the observed performance decline further underscores the importance of data augmentation in mitigating the data sparsity problem.

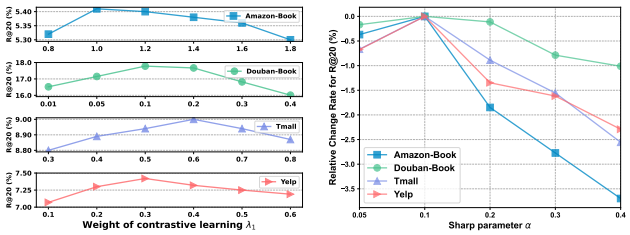
3.3.2 ***Hyperparameter Sensitivities.*** Most of MixRec's parameters are kept at the default settings (see Section 3.1.3 for details). Here we focus on two parameters λ_1 and α. Their performance variations on four datasets are shown in Fig. 5.

For the weight of contrastive losses, λ_1 shows different trends on four datasets, which is mainly due to the characteristics of the datasets themselves. For dense datasets, λ_1 usually takes smaller values (*e.g.*, Douban-Book); for sparse datasets, λ_1 takes larger values (*e.g.*, Tmall and Amazon-Book). The optimal values on Amazon-Book, Douban-book, Tmall, and Yelp datasets are 1.0, 0.1, 0.6 and 0.3 respectively.

For the shape parameter α, all datasets then show consistency. Specifically, in all cases, $\alpha = 0.1$ fetches the best recommendation performance. In other cases, too large a α will significantly degrade performance, indicating that the newly constructed sample

Table 6: Ablation studies of MixRec on Yelp, Amazon-Book, Tmall, and Douban-Book datasets.

	Yelp		Amazon-Book		Tmall		Douban-Book	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
w/o DMCL (user) (Eq. 9)	0.0676	0.0556	0.0484	0.0385	0.0826	0.0630	0.1566	0.1451
w/o DMCL (item) (Eq. 10)	0.0652	0.0540	0.0436	0.0342	0.0815	0.0608	0.1593	0.1473
w/o IM (Eq. 3)	0.0731	0.0605	0.0504	0.0403	0.0872	0.0665	0.1716	0.1628
w/o DM (Eq. 4)	0.0721	0.0597	0.0511	0.0409	0.0868	0.0664	0.1720	0.1639
MixRec	**0.0740**	**0.0612**	**0.0541**	**0.0433**	**0.0900**	**0.0686**	**0.1778**	**0.1712**



(a) λ_1 (b) α

Figure 5: Hyper-parameter Sensitivities for (a) the weight of loss λ_1 and (b) shape parameter α on four datasets.

is too corrupted to be considered a positive sample of the original view. Since α shows consistency across multiple datasets, we can set it to 0.1 by default without additional adjustment. Given that the shape parameter $\alpha = 0.1$, the number of GCN layers $L = 3$, the temperature coefficient $\tau = 0.2$, and the regularization coefficient $\lambda = 1e^{-4}$ are kept at their default values regardless of the datasets, **the only hyper-parameter MixRec can adjust is the contrastive coefficient λ_1.**

4 RELATED WORK

General Recommendation General recommendation is a branch of recommender systems that focuses solely on user-item interactions [21]. In this context, implicit feedback is widely used due to its easy accessibility [8]. However, improving the accuracy of greneral recommender system is challenging because of the sparsity of implicit feedback and the lack of semantic richness without auxiliary feature information [13, 41]. Early work focuses on matrix factorization [20], which eventually led to the introduction of neural networks to significantly enhance the model's learning capacity and generalization [8]. With the rise of graph neural networks, researchers have begun abstracting historical interactions into bipartite graph to model high-order relationships between users and items [26, 33]. From the graph perspective, traditional vector-based and neural network-based approaches can only capture first-order interactions, which limits their recommendation performance. Among these novel efforts, LightGCN [7] has been widely adopted in subsequent research due to its ease of deployment and has replaced matrix factorization as a foundational model. The success of graph-based methods is evident not only in general recommendation scenario but also in other recommendation branches, including social recommendation [42], knowledge graph recommendation [27], and multimodal recommendation [1], etc.

Self-supervised Learning for Recommendation Despite its considerable growth and a series of achievements, general recommendation still suffers from the sparsity of interaction data. Therefore, self-supervised learning is introduced into recommender systems to alleviate the data sparsity problem by constructing auxiliary tasks to provide additional supervision signals for the main recommendation task. In general, self-supervised learning can be broadly categorized into generative and contrastive models [18]. The former aims to model the distribution of the user community and reconstruct the complete interaction at a probabilistic level, as seen in models like Mult-VAE [15] and DiffRec [25]. The latter leverages data augmentation techniques to maximize mutual information between different views of the same sample. Earlier work focuses on modifying the original input data [2], such as in SGL [29], which randomly masks edges or nodes on the interaction graph. More recent work has shifted its focus toward finding new views within the feature space [36]. Examples include introducing noise for representations [32, 35], or finding semantic neighbors through clustering [16, 32], attention [19, 40], or hierarchical mechanisms [6]. Studies like DirectAU [22, 31, 38] revisit contrastive learning from the perspectives of alignment and uniformity [24]. SCCF [30], on the other hand, seeks to integrate graph convolution and contrastive learning into a unified framework. However, existing data augmentation strategies often suffer from high complexity and lack flexibility. More importantly, they typically measure the mutual information of a sample pair from only one perspective, failing to fully utilize the newly generated samples. MixRec is contrary to the design philosophy of pioneering works. Specifically, MixRec not only constructs richer new views with linear complexity but also maximizes the use of these samples through dual-mixing, enhancing its ability to support recommendation.

5 CONCLUSION

In this paper, we revisited the data sparsity problem entrenched in general recommendation and presented MixRec, an end-to-end dual mixing recommendation framework, which contains individual mixing and collective mixing for data augmentation. Specifically, individual mixing aims to construct new samples that are unique to the original inputs, while collective mixing considers the overall group perspective, creating new samples that represent the collective behavior of all users. Furthermore, we proposed dual-mixing contrastive learning to fully leverage all available sample pairs, maximizing the supervision signals provided for the recommendation task. we conducted extensive experiments on four real-world datasets and verified the effectiveness of MixRec.

References

[1] Haoyue Bai, Le Wu, Min Hou, Miaomiao Cai, Zhuangzhuang He, Yuyang Zhou, Richang Hong, and Meng Wang. 2024. Multimodality Invariant Learning for Multimedia-Based New Item Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 677–686.

[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 (ICLR)*.

[3] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[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 (ICML)*. 1597–1607.

[5] 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 (ICAIS)*. 249–256.

[6] Wei He, Guohao Sun, Jinhu Lu, and Xiu Susie Fang. 2023. Candidate-aware Graph Contrastive Learning for Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1670–1679.

[7] 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.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. 173–182.

[9] Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. 2021. Mixgcf: An improved training method for graph neural network-based recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 665–674.

[10] Sungnyun Kim, Gihun Lee, Sangmin Bae, and Se-Young Yun. 2020. Mixco: Mix-up contrastive learning for visual representation. *arXiv preprint arXiv:2010.06300* (2020).

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[13] Honghao Li, Lei Sang, Yi Zhang, and Yiwen Zhang. 2024. SimCEN: Simple Contrast-enhanced Network for CTR Prediction. In *Proceedings of the 32th ACM International Conference on Multimedia*.

[14] Sihang Li, Xiang Wang, An Zhang, Yingxin Wu, Xiangnan He, and Tat-Seng Chua. 2022. Let Invariant Rationale Discovery Inspire Graph Contrastive Learning. In *International Conference on Machine Learning*. PMLR, 13052–13065.

[15] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW)*. 689–698.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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 (UAI)*. 452–461.

[21] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. Springer, 1–35.

[22] 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.

[23] Feng Wang and Huaping Liu. 2021. Understanding the Behaviour of Contrastive Loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2495–2504.

[24] Tongzhou Wang and Phillip Isola. 2020. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *International Conference on Machine Learning (ICML)*. 9929–9939.

[25] Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. 2023. Diffusion Recommender Model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 832–841.

[26] 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.

[27] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web Conference 2021*. 878–887.

[28] Yu Wang, Yuying Zhao, Yi Zhang, and Tyler Derr. 2023. Collaboration-aware graph convolutional network for recommender systems. In *Proceedings of the ACM Web Conference 2023*. 91–101.

[29] 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 (SIGIR)*. 726–735.

[30] Yihong Wu, Le Zhang, Fengran Mo, Tianyu Zhu, Weizhi Ma, and Jian-Yun Nie. 2024. Unifying Graph Convolution and Contrastive Learning in Collaborative Filtering. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3425–3436.

[31] Liangwei Yang, Zhiwei Liu, Chen Wang, Mingdai Yang, Xiaolong Liu, Jing Ma, and Philip S Yu. 2023. Graph-based alignment and uniformity for recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 4395–4399.

[32] Yonghui Yang, Zhengwei Wu, Le Wu, Kun Zhang, Richang Hong, Zhiqiang Zhang, Jun Zhou, and Meng Wang. 2023. Generative-Contrastive Graph Learning for Recommendation. In *Proceedings of the 46th international ACM SIGIR Conference on Research and Development in Information Retrieval*. 1117–1126.

[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[34] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph Contrastive Learning with Augmentations. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5812–5823.

[35] 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.

[36] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2023. Self-supervised learning for recommender systems: A survey. *IEEE Transactions on Knowledge and Data Engineering* 36, 1 (2023), 335–355.

[37] An Zhang, Leheng Sheng, Zhibo Cai, Xiang Wang, and Tat-Seng Chua. 2023. Empowering Collaborative Filtering with Principled Adversarial Contrastive Loss. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[38] 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.

[39] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*.

[40] 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.

[41] Yi Zhang, Yiwen Zhang, Dengcheng Yan, Shuiguang Deng, and Yun Yang. 2023. Revisiting Graph-based Recommender Systems from the Perspective of Variational Auto-encoder. *ACM Transactions on Information Systems (TOIS)* 41, 3 (2023), 1–28.

[42] Yi Zhang, Yiwen Zhang, Yuchuan Zhao, Shuiguang Deng, and Yun Yang. 2024. Dual Variational Graph Reconstruction Learning for Social Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 36, 11 (2024), 6002–6015.

[43] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2021. Disentangling user interest and conformity for recommendation with causal embedding. In *Proceedings of the Web Conference 2021*. 2980–2991.