■ Appendixes

4.1 Model Analyses

4.1.1 Time Complexity Analyses

We provide analyses about two additional modules of our proposed *DiGCDR* as follows:

(1) Analysis of Graph Construction: We construct the global user–user and item–item graphs by computing Jaccard similarities over co-neighborhoods on the global user–item bipartite graph and keeping edges with sim $\geq \eta$. Using an inverted-index style enumeration that only visits co-occurring pairs, the overall construction cost is $O(|E|\,\rho)$, where $|E|$ is the number of observed interactions and ρ is the average degree. Also, this is a one-time offline preprocessing step.

(2) Analysis of HSIC Constraint: With batch size B, the linear kernel HSIC computation complexity is $O(Bd^2)$, where d is the dimensionality of the representations, which is typically quite small (e.g., 32, 64).

4.1.2 Connections between HSIC and disentanglement

A central task in contrastive CDR is to mitigate negative transfer, and a key pathway to achieve this is through representation disentanglement. We first devise a Collaborative Neighborhood Global Graph Construction module that extracts high-order, globally shared relations from the unified interactions while filtering out weak or irrelevant relations. Second, we propose a novel mutual information–based disentanglement objective, where InfoNCE promotes cross-domain alignment of shared representations, and HSIC enforces explicit statistical independence between shared and domain-specific representations. Specifically, HSIC measures statistical dependence, and HSIC=0 *if and only if* the variables are independent. When using characteristic kernels (e.g., RBF), this is equivalent to enforcing independence in a Reproducing Kernel Hilbert Space (RKHS), capable of capturing complex, non-linear dependencies—a significant advantage over simpler metrics like correlation. Please note that, HSIC is also computationally efficient in practice (mini-batch, low-dimensional computation with $O(Bd^2)$), yielding reliable and lightweight separation.

4.2 Experiment details

4.2.1 Dataset

Table 2 Dataset Statistics

Dataset	Domain	#Users	#Items	#Interactions	Density
Amazon	Sport	4,998	20,845	54,321	0.052%
	Phone	4,998	13,655	46,430	0.068%
	Sport	9,928	30,796	100,938	0.033%
	Cloth	9,928	39,008	95,369	0.024%
	Elec	15,761	51,447	224,689	0.027%
	Cloth	15,761	48,781	133,609	0.017%

To ensure a fair comparison with existing methods, four real-world benchmark datasets from Amazon are adopted in this work: *Elec*, *Phone*, *Sport*, and *Cloth*. The overall experimental settings follow those of BiTGCF [3], where the datasets are preprocessed to construct three cross-domain recommendation (CDR) scenarios: Task 1: *Sport & Phone*, Task 2: *Sport & Cloth*, and Task 3: *Elec & Cloth*. For the data in these three tasks, we first transform them into implicit data, where each entry is marked as 0 or 1, indicating whether the user has rated the item. Then, we filter the datasets to retain users with number of ratings greater than 5 and items with number of ratings greater than 10, and extract the overlapping users in both domains. Since the test set processed by BiTGCF contains a large number of cold-start items as ground-truth labels, which may lead to improper evaluation, we exclude those cold-start items from the test set in our experiments. The detailed statistics of the datasets are shown in Table 2.

4.2.2 Evaluation Protocol and Metrics

This work focuses on evaluating recommendation accuracy in a dual-domain setting, aligning with the task of dual-objective cross-domain intra-domain recommendation. We adopt the widely used leave-one-out protocol for performance evaluation. To ensure fair comparison across all methods, we employ a full-ranking strategy: for each user, the ground-truth item is ranked among all unobserved candidates.

We report Top-K performance (K=20) using two standard ranking-based metrics: **Recall** and **Normalized Discounted Cumulative Gain (NDCG)**. Higher values of these metrics indicate better recommendation accuracy.

Recall measures the proportion of relevant items that are successfully retrieved within the top-K recommendation list, and is defined as:

$$\text{Recall@}K = \frac{|Rec_u(K) \cap Rel_u|}{|Rel_u|} \tag{18}$$

The Normalized Discounted Cumulative Gain (NDCG) takes into account the positions of relevant items in the ranked list, giving higher importance to items appearing earlier.

First, the Discounted Cumulative Gain (DCG) is defined as:

$$\text{DCG@}K = \sum_{i=1}^{K} \frac{2^{r_{u,i}} - 1}{\log_2(i+1)} \tag{19}$$

The ideal DCG (IDCG) is defined as:

$$\text{IDCG@}K = \sum_{i=1}^{\min(K,|Rel_u|)} \frac{1}{\log_2(i+1)} \tag{20}$$

Then, the Normalized Discounted Cumulative Gain (NDCG) is computed as:

$$\text{NDCG@}K = \frac{\text{DCG@}K}{\text{IDCG@}K} \tag{21}$$

The symbols used in the above formulas are defined as follows:

- $Rec_u(K)$: the set of top-K items recommended to user u by the model;
- Rel_u: the set of items that user u has actually interacted with (i.e., relevant items);
- $rel_{u,i}$: the relevance score between user u and the item ranked at position i in the recommendation list (1 if relevant, 0 otherwise).

4.2.3 Implementation and hyperparameter setting

All experiments are implemented in PyTorch and deployed on an Nvidia RTX 3090 GPU (24GB). To ensure fairness, we standardize the latent dimension to $d = 64$ in all methods. The learning rate is fixed at 0.001 and training batch size is set to 2048. We adopt a uniform negative sampling strategy, where each positive instance is paired with one negative instance for all models during training. All layer parameters are initialized with the Xavier normal as provided by PyTorch. For GNN-based methods, the number of graph convolutional layers is selected from $\{2, 3\}$, and the final node embeddings are obtained through mean pooling across layers. Other important hyperparameter configurations are discussed in the Parameter Sensitivity Analysis section.

4.2.4 Competitors

We compare our proposed *DiGCDR* with representative state-of-the-art methods from both single-domain and cross-domain settings.

Single-Domain Baselines:

- MF is a classical matrix factorization model that represents users and items using latent factors.
- LightGCN [6] is a widely used graph-based recommendation model that captures user-item collaborative signals via a simplified graph convolutional network. It has demonstrated strong performance in top-K recommendation tasks.
- SimGCL [7] introduces a contrastive learning framework without requiring explicit graph augmentations, aiming to enhance recommendation effectiveness.

Cross-Domain Baselines:

- LightGCN-all is a straightforward extension of LightGCN that shares embeddings of overlapped users or items across two domains and leverages global collaborative signals during training. However, it does not address the issue of negative transfer.
- PTUPCDR+ [8] learns a meta-network over user feature embeddings to generate personalized mapping functions for transferring user preferences. In our dual-target domain setting, we adapt it to an end-to-end training scheme and introduce two mapping functions to enable bidirectional knowledge transfer between source and target domains.
- BITGCF [3] proposes a feature transfer layer that fuses domain-specific features within the graph convolution module, facilitating cross-domain information transfer.
- DisenCDR [4] presents a disentangled representation learning approach for cross-domain recommendation. It utilizes a variational autoencoder framework to decouple user interests into shared and domain-specific components.
- DRLCDR [9] leverages a conditional variational bipartite graph encoder to learn domain-specific and domain-conditional representations separately, employing decouple and bridge losses to ensure disentanglement and cross-domain consistency.
- DCCDR [5] combines graph convolution with contrastive learning, employing inter-domain contrastive constraints to better distinguish between shared and domain-specific information, thereby enhancing cross-domain recommendation performance.

4.3 Parameter Sensitivity Analysis

Table 3 Effect of Different Similarity Thresholds η on Recommendation Performance (NDCG@20)

Similarity Threshold η	Source Domain (Elec)	Target Domain (Cloth)
0.60	0.0210	0.0129
0.65	0.0217	0.0136
0.70	0.0236	0.0149
0.75	**0.0243**	**0.0152**
0.80	0.0230	0.0142

To further investigate the impact of key hyperparameters on the recommendation performance, we conduct a series of sensitivity experiments on Task 3. The evaluation metric is NDCG@20, and the performance on both the source domain (*Elec*) and the target domain (*Cloth*) is reported. The experimental results are shown in Table 3 and Figure 3.

- **Similarity Threshold η for Neighborhood Graph Construction:** In the Collaborative Neighborhood Global Graph Construction (CNGGC) of *DiGCDR* , the similarity threshold η determines whether an edge is established between users, thus critically influencing the structure of the global user–user graph and the overall model performance. To evaluate the sensitivity of this parameter, we test different thresholds $\eta \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ on Task 3. The results in Table 3 show that as η increases, the model performance on both domains first improves and then degrades. The best results are achieved when $\eta = 0.8$, indicating that appropriately filtering noisy edges while retaining high-quality user relationships is crucial for learning shared representations. When η is too small (e.g., 0.5 or 0.6), too many weakly correlated edges are included, introducing noise that hinders effective collaborative modeling. Conversely, when η is too large (e.g., 0.9), the number of edges becomes too sparse, resulting in insufficient structural information and reduced expressive power of the global graph. Therefore, setting a proper similarity threshold effectively balances graph density and quality, thereby improving recommendation performance. These findings verify the importance of η and highlight that the quality of the graph structure in the CNGGC module significantly affects cross-domain global information modeling.
- **Temperature Coefficient τ for Contrastive Learning:** We test τ within $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, and the results show that the best performance is achieved at $\tau = 0.2$. When the temperature is either too large or too small, model performance drops. This indicates that an appropriate temperature coefficient helps the model learn more effective contrastive signals, leading to more precise alignment between domain-shared and global representations. Therefore, τ should be carefully tuned for different datasets.
- **Smoothing Parameter σ of the Kernel Function:** The parameter σ is tested within $\{0.1, 0.2, 0.3, 0.4, 0.5\}$, and the best results are obtained when $\sigma = 0.3$ for both the source and target domains. This parameter controls the smoothness of the kernel function in the HSIC-based correlation constraint, thereby

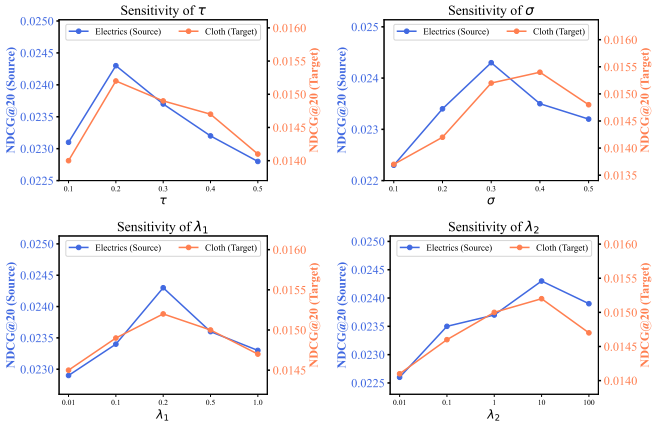

Fig. 3 Hyperparameter Sensitivity Analysis of *DiGCDR*.

influencing the separability among features. A small σ makes the kernel overly sensitive, acting only on very similar samples and ignoring global structure, while a large σ broadens the response range and weakens feature discriminability, reducing the model's ability to capture personalized characteristics. Hence, a moderate σ achieves a good trade-off between distinctiveness and generalization in the correlation constraint.

- **Weight of λ_1:** We vary λ_1 within $\{0.01, 0.1, 0.2, 0.5, 1.0\}$. The best performance is achieved at $\lambda_1 = 0.2$. Both too small and too large weights disrupt the balance among multi-task losses, causing either interference with the main objective or ineffective auxiliary supervision.
- **Weight of λ_2:** We tune λ_2 in $\{0.01, 0.1, 1, 10, 100\}$, and find that $\lambda_2 = 10$ yields the best performance. This suggests that a moderate level of disentanglement constraint facilitates better separation between shared and domain-specific representations without hindering the optimization of the main objective. Extremely small or large values of λ_2 impair the model's generalization ability.