DEBIASING MESSAGE PASSING TO MITIGATE POPULARITY BIAS IN GNN-BASED COLLABORATIVE FILTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Collaborative filtering (CF) models based on graph neural networks (GNNs) achieve strong performance in recommender systems by propagating user-item signals over interaction graphs. However, they are susceptible to popularity bias, since skewed interactions and repeated message passing across high-order neighborhoods amplify the influence of popular items while suppressing long-tail ones. Existing debiasing approaches, including re-weighting objectives, regularization, causal methods, and post-processing, are less effective in GNN-based settings because they do not directly counteract bias propagated through the aggregation process, and recent in-aggregation weighting methods often rely on static heuristics or unstable embedding estimates. We propose *Debiasing Popularity Amplification in Aggregation* (DPAA), a popularity debiasing framework for GNN-based CF that integrates adaptive, representation-aware interaction weighting and layer-wise weighting directly into message passing. DPAA assigns interaction-level weights from a representation-based popularity signal, stabilized by a smooth transition from pre-trained to evolving model embeddings during training. It further introduces a layer-wise weighting that amplifies higher-order neighborhoods, surfacing long-range interactions with diverse and underexposed items. Experiments on real-world and semi-synthetic datasets show that DPAA outperforms state-of-the-art popularity bias correction methods for GNN-based CF.

1 INTRODUCTION

Recommender systems have become a fundamental component of modern online platforms, helping users navigate large-scale information spaces and discover personalized content. These systems are deployed in diverse and complex digital ecosystems, including e-commerce and video-sharing platforms, social networks, and lifestyle applications (Chen et al., 2023). Collaborative filtering (CF) is a widely used recommendation paradigm that underpins many modern recommender systems. It infers user preferences directly from historical interaction data by leveraging patterns of similar user behavior (Lubos et al., 2023). In recent years, Graph Neural Networks (GNNs) have emerged as a powerful extension of traditional CF methods (He et al., 2020; Liu et al., 2021; Yu et al., 2023). By representing user-item interactions as a bipartite graph, GNN-based CF models learn expressive representations through iterative neighborhood aggregation, capturing high-order collaborative signals, and achieving state-of-the-art recommendation performance (He et al., 2020; Yu et al., 2023).

Recommender systems rely heavily on interaction data (e.g., clicks, views, watch events) to infer user preferences, but such data is often biased (Ai et al., 2018; Joachims et al., 2017), especially toward popular items (Zhou et al., 2023). User interactions are not only driven by true preferences but are also influenced by exposure mechanisms, as users are more likely to interact with frequently displayed items (Chen et al., 2023). Hence, observed interactions typically follow a long-tail distribution (Zhang et al., 2022), causing CF models trained on them to capture and reinforce popularity bias (Cañamares & Castells, 2018; Yao & Huang, 2017). This problem is further amplified in GNN-based CF, where message passing repeatedly propagates signals from high-degree popular items across local and higher-order neighborhoods (Zhou et al., 2023; Chen et al., 2024). Compared with sparsely connected niche items, popular items appear in many neighborhoods and exert disproportionate influence as their signals diffuse more broadly during message passing (Chen et al., 2020a; 2024). This repeated aggregation pulls user representations toward popularity-dominated regions,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

making popular items appear close to many users, increasing their predicted scores (Zhou et al., 2023), and causing them to be recommended more frequently. Such dynamics create a "rich-get-richer" feedback loop (Chaney et al., 2018), where popular items get more exposure while relevant long-tail items remain underexposed, degrading recommendation quality and user experience.

Existing popularity bias mitigation methods typically rely on inverse propensity weighting (IPW) to re-weight interactions (Ai et al., 2018; Gruson et al., 2019; Joachims et al., 2017), regularization (Boratto et al., 2021; Lin et al., 2025; Liu et al., 2023; Zhang et al., 2023a), causal modeling (Bonner & Vasile, 2018; Ning et al., 2024; Wang et al., 2021b; Wei et al., 2021; Zhao et al., 2022), or post-processing (Abdollahpouri et al., 2019; Chen et al., 2024; Ji et al., 2025; Zhu et al., 2021a) techniques. However, they do not address the repeated propagation and amplification of popularity bias during GNN aggregation (Chen et al., 2024; Kim et al., 2022; Zhou et al., 2023). To address this issue in GNN-based CF, recent studies (Kim et al., 2022; Zhou et al., 2023) incorporate weighting mechanisms directly into aggregation, assigning lower weights to interactions involving popular items, similar to propensity-based adjustments, to mitigate their disproportionate influence during message passing. For example, APDA (Zhou et al., 2023) estimates weights from the evolving user and item embeddings during training. However, early-epoch parameters can be noisy and unreliable due to rapid changes in weights and gradients (Frankle et al., 2020). Since these noisy weights are applied during aggregation, they can distort representation learning by propagating erroneous signals into node representations in later epochs. NAVIP (Kim et al., 2022) weights aggregation using static interaction counts and ignores embedding-space popularity signals, limiting its ability to mitigate popularity amplification in learned representations. TSP (Ji et al., 2025) mitigates topology-induced bias after training to improve long-tail performance, but post-hoc correction cannot address how biased graph structure and repeated propagation of popularity signals shape representation learning. DAP (Chen et al., 2024), a post-hoc GNN-based method, measures popularity using node degree, without considering interaction-level popularity effects encoded in the node embeddings.

We propose *Debiasing Popularity Amplification in Aggregation* (DPAA), a popularity debiasing method for GNN-based CF that directly applies bias-correcting weights to user-item interactions during GNN aggregation. These weights downweight interactions during aggregation that are more likely to reflect popularity-induced interactions and upweight interactions involving niche items. We introduce an embedding-aware interaction-strength metric that quantifies popularity-induced similarity in the embedding space at the interaction level and use it to assign weights during aggregation. To stabilize weight estimation in early training, the metric is initialized from a pre-trained model (e.g., the base recommendation model) and gradually transitions to the current model embeddings as they become reliable, yielding robust debiasing throughout training. To address popularity bias beyond the interaction level, we further introduce a layer-wise weighting mechanism in aggregation.

Our main contributions are summarized as follows:

- **Direct bias correction.** We propose DPAA, a unified popularity debiasing method for GNN-based CF that directly mitigates popularity amplification during message passing. DPAA combines stabilized adaptive embedding-aware bias-correcting weights with layer-wise weights to address both interaction-level and layer-level sources of popularity within a single method. DPAA controls whether interaction-level correction is applied at the initial or all layers, balancing under- and over-correction. Moreover, DPAA provides the flexibility to preserve node-specific information to mitigate over-smoothing. Therefore, DPAA differs from prior aggregation methods such as APDA and NAVIP in its weighting mechanism, stabilization of adaptive interaction weights, layer-wise weighting strategy, and control over possible over-correction and over-smoothing.
- **Theoretical characterization.** We show theoretically that our method DPAA reduces popularity dominance. More specifically, interaction-level weighting lowers the relative contribution of popular items depending on the severity of popularity bias, while layer-wise weighting increases the role of higher-order collaborative signals beyond shallow popularity-driven interactions.
- **Novel semi-synthetic data generation process.** Unlike prior studies that do not systematically evaluate debiasing methods under controlled popularity bias severity, we develop a semi-synthetic click data generation process that enables controlled variation of bias severity in the training data and allows us to evaluate how different methods perform for different levels of popularity bias.
- **Experiments.** Through experiments on real-world and semi-synthetic data, we show that DPAA outperforms existing methods from different methodological families of popularity bias correction across different bias levels and sparse scenarios, while remaining effective on the evaluated GNN-based CF backbones and improving performance for both popular and niche item groups.

2 PROBLEM FORMULATION

Problem setup. We consider a classical recommendation setting with users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and items $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$. For each user-item pair (u, i), let y_{ui} be a binary indicator where $y_{ui} = 1$ if user u has interacted with item i (e.g., through a click, rating, or watch event), and $y_{ui} = 0$ otherwise. These interactions form a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, with node set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ ($|\mathcal{V}| = |\mathcal{U}| + |\mathcal{I}|$), and edges in $\mathcal{A}$ connecting users to the items they have interacted with. The goal is to recommend the top-k most relevant items to each user that they have not interacted with.

Background. Graph convolutional networks (GCNs) have advanced representation learning in recommender systems (Kipf & Welling, 2017; Mao et al., 2021). GNN-based CF approaches (He et al., 2020; Lin et al., 2022b; Wang et al., 2019; 2020a; Wu et al., 2021; Yu et al., 2023; 2022) are built on neighborhood aggregation, where node representations are iteratively refined by aggregating information from their neighbors. A single graph convolution operation can be expressed as:

$$\mathbf{e}_u^{(l+1)} = f_{\text{aggregate}}\Big(\{\mathbf{e}_i^{(l)} : i \in \mathcal{N}_u \cup \{u\}\}\Big), \quad \mathbf{e}_i^{(l+1)} = f_{\text{aggregate}}\Big(\{\mathbf{e}_u^{(l)} : u \in \mathcal{N}_i \cup \{i\}\}\Big), \quad (1)$$

where $\mathbf{e}_u^{(l)}$ and $\mathbf{e}_i^{(l)}$ represent the embeddings of user u and item i after l propagation layers, while $\mathcal{N}_u = \{i \in \mathcal{I} : (u, i) \in \mathcal{A}\}$ and $\mathcal{N}_i = \{u \in \mathcal{U} : (u, i) \in \mathcal{A}\}$ denote the neighbor sets of user u and item i. The aggregation function $f_{\text{aggregate}}(\cdot)$ combines information from neighboring nodes' previous-layer embeddings to produce updated representations. The initial embeddings $\mathbf{e}^{(0)}$ serve as trainable ID-based representations for each user and item. After $(l + 1)$ rounds of message propagation, the user representation $\mathbf{e}_u^{(l+1)}$ or item representation $\mathbf{e}_i^{(l+1)}$ captures information from its $(l+1)$-hop neighbors. By stacking L propagation layers, a readout function $f_{\text{readout}}(\cdot)$ aggregates the representations from all layers to generate the final embedding:

$$\mathbf{e}_u = f_{\text{readout}}\Big([\mathbf{e}_u^{(0)}, \mathbf{e}_u^{(1)}, \ldots, \mathbf{e}_u^{(L)}]\Big), \quad \mathbf{e}_i = f_{\text{readout}}\Big([\mathbf{e}_i^{(0)}, \mathbf{e}_i^{(1)}, \ldots, \mathbf{e}_i^{(L)}]\Big). \quad (2)$$

Within GNN-based CF, LightGCN (He et al., 2020) is a widely used backbone model. The model uses the GCN message passing paradigm by eliminating feature transformations and non-linear activations, retaining only the mechanism of linear neighborhood aggregation. It uses a weighted-sum aggregator as the aggregation function which is defined as: $\mathbf{e}_u^{(l+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{d_u d_i}} \mathbf{e}_i^{(l)}$, $\mathbf{e}_i^{(l+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{d_i d_u}} \mathbf{e}_u^{(l)}$, where d_u and d_i denote the degrees of user u and item i. To get the final representation, LightGCN employs a mean-pooling readout function that aggregates embeddings from all $(L+1)$ layers: $\mathbf{e}_u = \frac{1}{L+1} \sum_{l=0}^{L} \mathbf{e}_u^{(l)}$, $\mathbf{e}_i = \frac{1}{L+1} \sum_{l=0}^{L} \mathbf{e}_i^{(l)}$. The predicted score between a user u and an item i is computed as the inner product of their final embeddings: $\hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i$. LightGCN is optimized using the Bayesian Personalized Ranking (BPR) loss (Rendle et al., 2009). This is defined over triplets (u, i, j), where i is an interacted item for user u and j is not an interacted one, encouraging the model to predict i higher than j: $\mathcal{L}_{BPR} = \sum_{u, i \in \mathcal{N}_u, j \notin \mathcal{N}_u} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \rho \|\Theta\|_2^2$. Here, ρ is the regularization coefficient and Θ represents the current batch 0-th layer embeddings.

Problem statement. The goal of an unbiased GNN-based CF model is to learn node representations ($\mathbf{e}_u$ and $\mathbf{e}_i$) that are close to their optimal unbiased counterparts ($\mathbf{e}_u^{(*)}$ and $\mathbf{e}_i^{(*)}$). Unbiased representations are learned from interactions reflecting true preferences rather than bias factors. We can formalize the objective as minimizing the deviation between the model embeddings and their unbiased counterparts over all nodes $\mathcal{V}$: $\min_{\{\mathbf{e}_j\}_{j \in \mathcal{V}}} \sum_{j \in \mathcal{V}} \left\| \mathbf{e}_j - \mathbf{e}_j^{(*)} \right\|_2^2$. However, we cannot estimate the optimal unbiased embeddings because real-world data is biased. The challenge of debiasing is that we need to approximate unbiased representations from biased data. Our goal is to approximate unbiased embeddings indirectly by reducing the effect of bias during representation learning.

3 OUR METHOD: DPAA

In this section, we introduce DPAA (*Debiasing Popularity Amplification in Aggregation*), our proposed method for mitigating popularity bias in GNN-based CF. DPAA integrates adaptive bias-correcting weights directly into message passing. This is based on the insight that popularity bias is amplified during GNN aggregation. Therefore, effective mitigation requires intervening in the propagation stage. Formally, we modify the standard GNN message passing operator by introducing a weighted adjacency structure, where each user–item interaction is assigned a dynamic weight that

depends on both interaction-level and layer-level factors. This results in a reweighted aggregation operator that explicitly controls the propagation of popularity-biased signals during training.

3.1 Estimating weights for counteracting popularity amplification

3.1.1 Interaction-level weight estimation

In standard CF, the inner product between user and item embeddings measures user–item similarity and is used to rank items. The BPR objective encourages observed user–item pairs to have larger inner products than unobserved pairs. However, in popularity-biased data, observed positives are shaped by exposure. Popular items appear in many interactions, propagate their signals to a large number of users, and hence their embeddings become more similar to many user embeddings (Zhou et al., 2023). Thus, a high inner product between a popular item's embedding and a user's embedding may reflect popularity-driven interactions (Zhou et al., 2023). Based on this, we treat user–item similarity as a proxy for interaction strength, where highly similar user-item pairs are more likely to involve popular items. Note that symmetric normalization in graph aggregation (e.g., degree normalization in LightGCN) addresses graph structural imbalance but cannot alone prevent popular items from obtaining large inner products, as their many connections keep the sum of aggregated node degrees large, increasing the inner product (Zhou et al., 2023).

Definition 1 (Current Model Inverse Interaction Weight). *Given the user and item embeddings $e_{u,t}^{(l)}$ and $e_{i,t}^{(l)}$ from the current model c at layer l and epoch t, we define the current model inverse interaction weight $r_{ui,c,t}^{(l)}$ as the complement of their L_2-normalized inner product:*

$$r_{ui,c,t}^{(l)} = r_{iu,c,t}^{(l)} = 1 - \frac{e_{u,t}^{(l)\top} e_{i,t}^{(l)}}{\|e_{u,t}^{(l)}\|_2 \, \|e_{i,t}^{(l)}\|_2}. \tag{3}$$

Instead of the raw inner product, we use cosine similarity, which incorporates L_2 normalization to remove the effect of embedding magnitude. $r_{ui,c,t}^{(l)}$ assigns weights inversely proportional to the interaction strength estimated from the current model embeddings, thereby giving smaller weights to interactions involving popular items and larger weights to interactions involving long-tail items.

A naive approach that relies solely on current embeddings can be unstable, especially in early training, where embeddings change rapidly and are noisy (Frankle et al., 2020). Applying weights estimated from such embeddings during aggregation may impact the learning process and embedding updates during backpropagation. To address this, we adopt an interaction-strength estimation strategy that uses embeddings from a pre-trained model. We use the embeddings of a pre-trained model, such as the base recommendation model that generated the current interaction data. This model provides stable embeddings for estimating weights at the early stages of training. These weights are pre-computed and remain fixed for each interaction at a specific layer l throughout training.

Definition 2 (Pre-trained Model Inverse Interaction Weight). *Given the user and item embeddings $e_{u,m}^{(l)}$ and $e_{i,m}^{(l)}$ from the pre-trained model m at layer l, we define the pre-trained model inverse interaction weight $r_{ui,m}^{(l)}$ as the complement of their L_2-normalized inner product:*

$$r_{ui,m}^{(l)} = r_{iu,m}^{(l)} = 1 - \frac{e_{u,m}^{(l)\top} e_{i,m}^{(l)}}{\|e_{u,m}^{(l)}\|_2 \, \|e_{i,m}^{(l)}\|_2}. \tag{4}$$

To obtain an interaction-strength estimate that is reliable early in training and adaptive later, we combine the inverse interaction weights from the pre-trained and current models using a time-dependent convex combination. The pre-trained estimate provides stable signals in early training, while the current evolving model estimate captures evolving representations as learning progresses. This formulation ensures a smooth transition between the pre-trained and current model estimates.

Definition 3 (Inverse Interaction Weight). *Given the current model inverse interaction weight $r_{ui,c,t}^{(l)}$ and the pre-trained model inverse interaction weight $r_{ui,m}^{(l)}$ at training epoch t, we define the inverse interaction weight (IIW) as:*

$$r_{ui,t}^{(l)} = r_{iu,t}^{(l)} = \beta_t \, r_{ui,m}^{(l)} + (1 - \beta_t) \, r_{ui,c,t}^{(l)}, \tag{5}$$

where $\beta_t \in [0, 1]$ is an epoch-dependent coefficient that controls the relative contribution of the pre-trained and current model estimates during training.

The function is bounded in $[0, 1]$ and adapts continuously based on embedding stability, avoiding abrupt transitions. In early epochs, β_t is close to one, so IIW relies more on the pre-trained signal. As training progresses, β_t decreases, allowing the current model to increasingly determine the weights. To determine this transition, we measure the stability of current model embeddings using the average embedding change between consecutive epochs: $\Delta_t = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \|\mathbf{e}_{v,t} - \mathbf{e}_{v,t-1}\|_2$, where $\mathbf{e}_{v,t}$ and $\mathbf{e}_{v,t-1}$ denote the embeddings of node v at epochs t and $t-1$, respectively. A larger Δ_t indicates unstable embeddings, while a smaller Δ_t indicates more stable representations of the current model. We then define $\beta_t = \frac{\Delta_t}{\Delta_t + C}$, where $C \geq 0$ controls the sensitivity of the transition. With this formulation, large embedding changes yield $\beta_t \to 1$, making $r^{(l)}_{ui,t}$ rely mainly on the pre-trained estimate $r^{(l)}_{ui,m}$. As the embeddings stabilize and Δ_t decreases, $\beta_t \to 0$, so the current-model estimate $r^{(l)}_{ui,c,t}$ becomes dominant. This adaptive transition prevents noisy early-stage embedding fluctuations, as in (Zhou et al., 2023), while preserving model-adaptive weighting in later training.

We provide empirical evidence that interactions with more popular items generally exhibit higher user–item similarity. By grouping items into different popularity quartiles, we observe that more popular item groups generally exhibit higher user–item cosine similarity in GNN-based CF models on real-world datasets (Figure 5 in the Appendix), potentially favoring popular items over tail items in recommendations from GNN-based CF models. This supports our IIW mechanism, which uses the complement of cosine similarity to assign smaller weights during aggregation to interactions with higher user-item similarity. A detailed analysis is provided in Appendix A.3.3.

3.1.2 Layer-level weight estimation

To address popularity bias beyond IIW, we introduce a layer-wise weighting mechanism that controls the contribution of different GNN layers during message passing. Lower layers mainly capture immediate user–item interactions, which are often dominated by frequently displayed items. In contrast, higher layers aggregate information from multi-hop neighbors, allowing user representations to incorporate signals from a more diverse set of items connected through shared collaborative contexts, including potentially relevant long-tail items. This enriches the representations beyond immediate popularity-driven interactions. Although deeper layers can carry popularity signals, their representations aggregate signals through multi-hop paths, making them less confined to a user's immediate popularity-driven interactions. Therefore, we can assign larger weights to deeper layers, giving more importance to higher-order item signals and reducing shallow popularity dominance.

Definition 4 (Layer-wise weighting). *We formalize the layer-wise weight (LW) as a function of the layer index. The weight assigned to layer l is defined as:*

$$\lambda_l = (l+1)^\eta, \tag{6}$$

where $\eta \geq 0$ is a hyperparameter controlling the strength of emphasis on different layers.

In practice, λ_l can be normalized across layers to maintain scale consistency and prevent domination by deep layers. When $\eta = 0$, all layers contribute equally, resulting in uniform aggregation. When $\eta > 0$, deeper layers receive larger weights, giving stronger emphasis to higher-order neighbors. The hyperparameter η controls the contribution of deeper propagation, encouraging reliance on higher-order neighborhoods when beneficial while limiting over-reliance when they degrade performance.

The complete weighting mechanism unifies IIW and LW into a single formulation. The resulting weight $w^{(l)}_{ui,t}$ for the interaction between user u and item i at layer l for epoch t can be written as:

$$w^{(l)}_{ui,t} = \lambda_l \left[(1-\gamma) r^{(l)}_{ui,t} + \gamma \left(\mathbb{I}(l=0) r^{(l)}_{ui,t} + \mathbb{I}(l>0) \right) \right]. \tag{7}$$

The parameter $\gamma \in \{0, 1\}$ controls where IIW is applied, and $\mathbb{I}(\cdot)$ is an indicator function. When $\gamma = 0$, IIW is applied at every propagation layer. When $\gamma = 1$, IIW is applied only at the initial layer, while higher layers rely solely on LW. This design allows representations corrected by IIW at the initial layer to propagate to higher layers, while providing flexibility to apply IIW across all layers when deeper layers also require IIW-correction. This helps balance potential under-correction when IIW is applied only at the initial layer and over-correction when it is applied across all layers.

3.2 Debiasing popularity amplification in aggregation

Embeddings at $(l+1)^{th}$ layer at training epoch t can be written using the weight as follows:

$$\mathbf{e}^{(l+1)}_{u,t} = f_{\text{aggregate}}\Big(\{w^{(l)}_{ui,t}\mathbf{e}^{(l)}_{i,t} : i \in \mathcal{N}_u \cup \{u\}\}\Big); \mathbf{e}^{(l+1)}_{i,t} = f_{\text{aggregate}}\Big(\{w^{(l)}_{iu,t}\mathbf{e}^{(l)}_{u,t} : u \in \mathcal{N}_i \cup \{i\}\}\Big).$$

Figure 3 in the Appendix provides an overview of DPAA's weighted message passing mechanism.

When multiple GNN layers are stacked, over-smoothing (Chen et al., 2020a) can make node representations similar, reducing local structural distinctions among nodes. Since IIW is computed from user–item similarity, excessive smoothing can reduce variation among IIW values. In extreme case, when all embeddings become nearly identical, IIW values also become nearly uniform, diminishing the bias-correcting capability of IIW. Moreover, assigning larger weights to deeper layers through LW can sometimes exacerbate over-smoothing. To mitigate these, following (Chen et al., 2020b), we use an *initial residual connection* δ that preserves node-specific information at each layer:

$$\hat{\boldsymbol{e}}_{u,t}^{(l)} = \boldsymbol{e}_{u,t}^{(l)} + \delta \boldsymbol{e}_{u,t}^{(0)}, \quad \hat{\boldsymbol{e}}_{i,t}^{(l)} = \boldsymbol{e}_{i,t}^{(l)} + \delta \boldsymbol{e}_{i,t}^{(0)}. \tag{8}$$

3.3 Theoretical analysis

We provide theoretical analyses explaining how the two main components of DPAA reduce popularity dominance. First, IIW reduces the relative influence of popular items, applying larger correction when popularity bias is more pronounced. Second, LW increases the role of higher-order signals by giving them larger contributions to embedding updates during backpropagation, allowing the model to incorporate more diverse and relevant items beyond popularity-driven interactions.

Relative message passing contribution reduction via IIW. We analyze the effect of applying IIW during message passing. Setting $\lambda_l = 1$ in equation 3.2, the IIW-only aggregation from item i to user u at layer $(l+1)$ becomes $\mathbf{e}_{u,t}^{(l+1)} = \sum_{i \in \mathcal{N}(u)} r_{ui,t}^{(l)} \, \mathbf{e}_{i,t}^{(l)}$. Since $r_{ui,t}^{(l)}$ is the complement of the normalized user-item similarity, popular items, which become broadly similar with many users through repeated propagation, receive smaller weights and contribute less to aggregation. We define the message passing contribution of item i as the total weighted magnitude of the messages it sends to its neighbors: $\mathcal{I}_{\mathrm{IIW}_i}^{(l)} = \sum_{u \in \mathcal{N}(i)} r_{ui,t}^{(l)} \left\| \mathbf{e}_{i,t}^{(l)} \right\|_2 = d_i \, \bar{r}_{i,t}^{(l)} \left\| \mathbf{e}_{i,t}^{(l)} \right\|_2$, where $d_i = |\mathcal{N}(i)|$ is the item degree and $\bar{r}_{i,t}^{(l)} = \frac{1}{d_i} \sum_{u \in \mathcal{N}(i)} r_{ui,t}^{(l)}$ is its average IIW. The standard GNN contribution is obtained by setting $r_{ui,t}^{(l)} = 1$, giving $\mathcal{I}_i^{(l)} = d_i \|\mathbf{e}_{i,t}^{(l)}\|_2$. Because popular items are repeatedly propagated through many users, they become more broadly similar with user embeddings than long-tail items (Zhou et al., 2023). Thus, for any popular item p and long-tail item q, we have $\bar{r}_{p,t}^{(l)} < \bar{r}_{q,t}^{(l)}$, with $\bar{r}_{q,t}^{(l)} > 0$.

Lemma 3.1 (Popularity influence reduction.) *For any popular item p and long-tail item q with $d_p > d_q$ and $\bar{r}_{p,t}^{(l)} < \bar{r}_{q,t}^{(l)}$, IIW reduces the relative message-passing contribution of the popular item by the following closed-form positive reduction:*

$$\frac{\mathcal{I}_p^{(l)}}{\mathcal{I}_q^{(l)}} - \frac{\mathcal{I}_{\mathrm{IIW}_p}^{(l)}}{\mathcal{I}_{\mathrm{IIW}_q}^{(l)}} = \left(1 - \frac{\bar{r}_{p,t}^{(l)}}{\bar{r}_{q,t}^{(l)}}\right) \cdot \frac{\mathcal{I}_p^{(l)}}{\mathcal{I}_q^{(l)}} > 0. \tag{9}$$

We provide the proof in Appendix A.2.1. Lemma 3.1 has two important implications:

- *First,* the reduction depends on the gap between the average IIW values of popular and niche items. A larger gap, which can arise when popularity bias is severe, indicates a clearer embedding-space distinction between popular and long-tail items and leads to a larger reduction in popular-item dominance compared to standard GNN. Similarly, a smaller gap leads to a smaller reduction.
- *Second,* the reduction is amplified by the structural degree imbalance d_p/d_q, since $\mathcal{I}_p^{(l)}$ and $\mathcal{I}_q^{(l)}$ depend on item degrees. When popular items have many interactions and long-tail items have very few, this imbalance becomes large, so IIW applies a larger correction compared to the standard GNN. Thus, IIW adaptively mitigates bias based on the severity of popularity skew in the data.

Influence of LW on representation learning. We now analyze how the LW component of DPAA reshapes representation learning. In GNN-based CF, embeddings are updated through the gradient of the objective function, so applying LW during message propagation in the forward pass affects how the embeddings are updated during backpropagation. We analyze how LW influences the gradients of the loss with respect to the embeddings. Consider a training triple (u, i, j), where user u has interacted with item i but not with item j. Let $\phi_v(\mathbf{e}^{(0)})$ be a function of the initial embeddings, and let $\psi_\ell(\mathbf{h}_v^{(\ell)})$ be a function of the layer-ℓ neighborhood aggregation of node v.

Lemma 3.2 (Layer-wise influence on representation learning.) *Applying layer-wise weighting during message passing, the gradient of the objective function $\mathcal{L}_{uij}$ with respect to embedding $\mathbf{e}_v$*

for any node $v \in \{u, i, j\}$ *satisfies:*

$$\frac{\partial \mathcal{L}_{uij}}{\partial \mathbf{e}_v} \propto \phi_v(\mathbf{e}^{(0)}) + \sum_{\ell=1}^{L} \ell^{\eta} \cdot \psi_\ell(\mathbf{h}_v^{(\ell)}). \tag{10}$$

We provide the proof in Appendix A.2.2. Lemma 3.2 implies the following:

- *First,* for any two layers ℓ_a and ℓ_b where $\ell_a < \ell_b$, the deeper layer receives a larger gradient coefficient $(\ell_a)^\eta < (\ell_b)^\eta$ when $\eta > 0$, whereas in a standard GNN without LW, all layers contribute equally (i.e., $\ell^\eta = 1$ for all ℓ when $\eta = 0$). LW therefore structurally amplifies gradient influence toward deeper layers, reducing reliance on shallow popularity-dominated updates and distributing representation learning across a broader set of items from higher-order neighborhoods.
- *Second,* as η increases, the ratio $(\ell_b)^\eta/(\ell_a)^\eta > 1$ grows, progressively amplifying the contribution of deeper layers to embedding updates. In scenarios where immediate interactions are heavily dominated by popular items, larger η can be particularly beneficial, as this shifts gradient influence away from popularity-skewed local interactions toward higher-order collaborative signals.

4 EXPERIMENTS

We evaluate our method DPAA by answering the following research questions:

- **RQ1:** How effective is DPAA relative to existing methods for mitigating popularity bias?
- **RQ2:** How robust is DPAA across different levels of popularity bias severity?
- **RQ3:** How does DPAA perform across popular and niche item groups, and is there a trade-off?
- RQ4: Does DPAA remain effective when applied to a different GNN-based CF backbone?

Code is available here: `https://anonymous.4open.science/r/DPAA`.

4.1 EXPERIMENTAL SETUP

Datasets and evaluation metrics. We conduct experiments on three real-world datasets: Coat (Schnabel et al., 2016), KuaiRec (Gao et al., 2022), and Yahoo! R3 (Marlin & Zemel, 2009). These datasets cover different domains: coat shopping, short-video recommendation, and music recommendation. We choose these datasets because they are among the few public datasets with unbiased test sets for unbiased evaluations. Coat and Yahoo! R3 provide test sets collected with random exposure, while the KuaiRec test set is collected with a near-full exposure setting. These datasets vary in scale, sparsity, and distributions, reflecting diverse real-world scenarios. Figure 4 in the Appendix shows that a small fraction of items accounts for 80% of training interactions in these datasets, highlighting the popularity skew in real-world data. We use Recall@k, NDCG@k, and HR@k as evaluation metrics. We report results over five independent runs with different random seeds. Detailed dataset statistics and distributions are provided in Appendices A.3.1 and A.3.2. Details on unbiased evaluation and ranking protocols are in Appendix A.3.4.

Semi-synthetic data generation with controlled popularity bias. A popularity bias correction method should remain effective with varying bias severity, but real-world datasets do not allow direct control over the degree of popularity. To the best of our knowledge, existing studies do not evaluate debiasing methods across different bias levels. To enable systematic evaluation, we develop a semi-synthetic click-data generation process using the unbiased KuaiRec test set, which is collected under full-exposure settings. We sample training interactions with a Zipfian distribution (Newman, 2005) and vary the skewness parameter from $s = 0$ to $s = 9$, where larger s indicates stronger popularity. We refer to s as the bias severity parameter. Details are provided in Appendix A.3.5.

Baselines. We compare DPAA with popularity bias correction methods from major debiasing families, including GNN aggregation re-weighting methods APDA (Zhou et al., 2023) and NAVIP (Kim et al., 2022), post-hoc embedding correction method DAP (Chen et al., 2024), causal-inference methods PPAC (Ning et al., 2024), MACR (Wei et al., 2021) and CVIB (Wang et al., 2020b), IPW-based method IPS (Gruson et al., 2019), and regularization-based method SAM-REG (Boratto et al., 2021). Since our study focuses on CF using only interaction data and no side information, we include baselines that are compatible with this setting. Unless stated otherwise, we use LightGCN as the backbone for our method and all baselines to ensure a fair comparison. We also use the more recent LightGCN++ (Lee et al., 2026) as the backbone for our method in RQ4 to validate its effectiveness with another GNN-based CF backbone. Detailed baseline descriptions and implementation details are provided in Appendix A.3.6 and Appendix A.3.7, respectively.

4.2 RESULTS

Overall performance comparison (RQ1). We evaluate whether DPAA outperforms existing popularity bias correction methods and show the results in Table 1. DPAA achieves the best performance on all datasets for Recall@20, NDCG@20, and HR@20. Compared with GNN aggregation-based baselines such as APDA and NAVIP, DPAA performs substantially better, suggesting the benefit of using stabilized IIW with LW. DPAA provides clear gains on Coat, which exhibits milder popularity skew. The largest improvements occur on KuaiRec, which exhibits severe popularity skew (Figure 4), suggesting that DPAA is especially effective when popularity strongly dominates the data. Some existing methods perform poorly on KuaiRec, likely due to its severe popularity skew, suggesting that they may be less suitable in such settings. On Yahoo! R3, DPAA outperforms all methods, although its gains are relatively small. This is likely due to the high sparsity of the dataset (Table 3), which makes further improvements more challenging than on the other datasets for all methods. The low variability in the results suggests that DPAA's improvements over the baselines are consistent. These results demonstrate DPAA's effectiveness with varying popularity skew and sparsity. We analyze parameters in Appendix A.4.1 and computational cost in Appendix A.4.4.

Table 1: Performance comparison of different methods. Bold and underline indicate the best and second-best results. Results are reported as mean $\pm$ std%. Improvement (%) denotes the relative gain of DPAA over the best baseline.

Methods	Coat			KuaiRec			Yahoo! R3		
	Recall	NDCG	HR	Recall	NDCG	HR	Recall	NDCG	HR
DPAA (Ours)	**0.2565**$_{\pm 0.71}$	**0.2065**$_{\pm 0.87}$	**0.6234**$_{\pm 1.45}$	**0.0349**$_{\pm 0.49}$	**0.0586**$_{\pm 0.29}$	**0.6201**$_{\pm 1.02}$	**0.1479**$_{\pm 0.09}$	**0.0659**$_{\pm 0.06}$	**0.2221**$_{\pm 0.16}$
APDA	0.1587$_{\pm 0.25}$	0.0745$_{\pm 0.51}$	0.4626$_{\pm 0.69}$	0.0113$_{\pm 0.62}$	0.0113$_{\pm 0.41}$	0.1082$_{\pm 1.22}$	0.1414$_{\pm 0.09}$	0.0644$_{\pm 0.60}$	0.2111$_{\pm 0.42}$
NAVIP	0.1255$_{\pm 0.34}$	0.0635$_{\pm 0.68}$	0.3808$_{\pm 0.86}$	0.0016$_{\pm 0.27}$	0.0090$_{\pm 0.21}$	0.1128$_{\pm 0.65}$	<u>0.1452</u>$_{\pm 0.51}$	<u>0.0645</u>$_{\pm 0.20}$	<u>0.2175</u>$_{\pm 0.61}$
DAP	0.1374$_{\pm 0.77}$	0.0742$_{\pm 1.02}$	0.4306$_{\pm 1.68}$	0.0222$_{\pm 0.72}$	0.0120$_{\pm 0.46}$	0.1858$_{\pm 1.12}$	0.1423$_{\pm 0.21}$	0.0640$_{\pm 0.18}$	0.2157$_{\pm 0.24}$
PPAC	0.1565$_{\pm 0.49}$	0.0759$_{\pm 0.61}$	0.4626$_{\pm 1.27}$	<u>0.0313</u>$_{\pm 0.23}$	<u>0.0474</u>$_{\pm 0.24}$	<u>0.3340</u>$_{\pm 0.76}$	0.1449$_{\pm 0.11}$	0.0641$_{\pm 0.09}$	0.2142$_{\pm 0.21}$
MACR	0.1002$_{\pm 0.21}$	0.0451$_{\pm 0.23}$	0.3345$_{\pm 0.54}$	0.0039$_{\pm 0.21}$	0.0041$_{\pm 0.19}$	0.0816$_{\pm 0.56}$	0.0095$_{\pm 0.02}$	0.0033$_{\pm 0.04}$	0.0190$_{\pm 0.12}$
CVIB	<u>0.2301</u>$_{\pm 0.59}$	0.1880$_{\pm 0.89}$	<u>0.5867</u>$_{\pm 1.89}$	0.0116$_{\pm 0.29}$	0.0134$_{\pm 0.20}$	0.1106$_{\pm 0.71}$	0.1435$_{\pm 0.23}$	0.0643$_{\pm 0.21}$	0.2127$_{\pm 0.41}$
IPS	0.2107$_{\pm 0.19}$	<u>0.1905</u>$_{\pm 0.23}$	0.5658$_{\pm 0.48}$	0.0009$_{\pm 0.23}$	0.0060$_{\pm 0.20}$	0.0575$_{\pm 0.49}$	0.1300$_{\pm 0.31}$	0.0578$_{\pm 0.19}$	0.1972$_{\pm 0.35}$
SAM-REG	0.1874$_{\pm 0.81}$	0.1488$_{\pm 0.78}$	0.5124$_{\pm 0.91}$	0.0015$_{\pm 0.29}$	0.0086$_{\pm 0.31}$	0.0830$_{\pm 0.71}$	0.1229$_{\pm 0.53}$	0.0551$_{\pm 0.30}$	0.1859$_{\pm 0.89}$
LightGCN	0.1301$_{\pm 0.31}$	0.0715$_{\pm 0.61}$	0.4093$_{\pm 0.82}$	0.0039$_{\pm 0.27}$	0.0029$_{\pm 0.21}$	0.0487$_{\pm 0.95}$	0.1411$_{\pm 0.23}$	0.0628$_{\pm 0.09}$	0.2134$_{\pm 0.34}$
Improvement (%)	+11.47%	+8.40%	+6.26%	+11.50%	+23.63%	+85.66%	+1.86%	+2.17%	+2.11%

Robustness under different popularity bias severity (RQ2). We vary the popularity bias severity s from 0 to 9 using the semi-synthetic click generation process on the KuaiRec dataset. As shown in Figure 1, DPAA outperforms all baselines except at $s = 1$, achieving modest improvements at low bias severity ($s = 0$–3) and larger gains as bias increases. When popularity bias is mild, LightGCN often outperforms most debiasing baselines, suggesting that some methods over-correct bias and hurt performance. DPAA remains competitive in these settings, indicating that it avoids over-correction. Although performance generally declines for all methods as bias increases, DPAA achieves larger gains over the baselines under severe bias, suggesting that it avoids under-correction. These results demonstrate DPAA's robustness across bias levels and are consistent with Lemma 3.1, which shows that the reduction in popularity dominance depends on bias severity. We also provide the results in Table 6 in the Appendix.

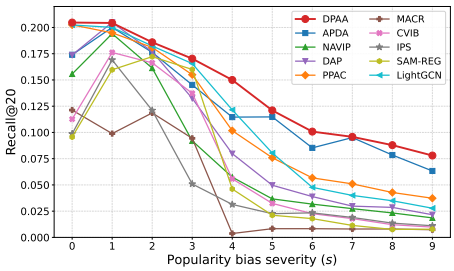


Figure 1: Performance comparison of different methods for Recall@20 on the KuaiRec dataset by varying levels of popularity bias severity (s).

Performance on popular and niche item groups (RQ3). RQ3 evaluates performance on popular and niche item groups, examining whether improving performance in one group affects the other. Following (Abdollahpouri et al., 2019), we sort items by interaction frequency and define the smallest set covering 80% of interactions as popular items, with the remaining items treated as long-tail ones (Figure 4). As shown in Table 2, DPAA consistently outperforms the baselines across both item groups. The low KuaiRec niche scores reflect its highly imbalanced tail distribution, where many niche items have few interactions and may not closely align with users' personalized preferences. As we perform unbiased evaluations, improvements in both groups show that DPAA does not simply shift recommendations from popular to niche items. It improves performance for both groups by reducing popularity amplification while not suppressing preference-relevant signals.

DPAA with a different GNN-based CF backbone (RQ4). We evaluate whether DPAA remains effective when used with another GNN-based CF backbone, such as LightGCN++ (Lee et al., 2026),

Table 2: Performance comparison on the popular and the niche item groups. Improvement (%) denotes the relative gain of DPAA over the best baseline.

Methods	Coat				KuaiRec				Yahoo! R3			
	Popular		Niche		Popular		Niche		Popular		Niche	
	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
DPAA (Ours)	**0.3274**	**0.2288**	**0.1202**	**0.0917**	**0.0360**	**0.0601**	**0.0002**	**0.0003**	**0.2467**	**0.1064**	**0.0191**	**0.0065**
APDA	0.2097	0.0935	0.0057	0.0019	0.0128	0.0098	0.0000	0.0000	0.2329	0.1038	0.0155	0.0057
NAVIP	0.1761	0.0818	0.0166	0.0053	0.0017	0.0064	0.0000	0.0000	0.2383	0.1038	0.0185	0.0064
DAP	0.1822	0.0870	0.0284	0.0090	0.0256	0.0134	0.0001	0.0001	0.1982	0.0864	0.0184	0.0060
PPAC	0.1779	0.0918	0.0523	0.0223	0.0331	0.0477	0.0000	0.0000	0.2361	0.1016	0.0152	0.0059
MACR	0.1141	0.0469	0.0555	0.0202	0.0045	0.0048	0.0000	0.0000	0.0003	0.0001	0.0183	0.0059
CVIB	0.2975	0.2102	0.1093	0.0828	0.0274	0.0299	0.0000	0.0000	0.2372	0.1043	0.0134	0.0057
IPS	0.2643	0.2057	0.1098	0.0901	0.0012	0.0059	0.0000	0.0000	0.2120	0.0923	0.0180	0.0058
SAM-REG	0.2242	0.1518	0.1093	0.0792	0.0018	0.0086	0.0001	0.0001	0.2096	0.0904	0.0066	0.0027
LightGCN	0.1905	0.0939	0.0010	0.0005	0.0042	0.0027	0.0000	0.0000	0.2341	0.1017	0.0101	0.0044
Improvement (%)	+10.05%	+8.85%	+9.47%	+1.78%	+8.76%	+26.00%	+100.0%	+200.0%	+3.52%	+2.01%	+3.24%	+1.56%

under varying levels of popularity bias. We apply DPAA to the LightGCN++ backbone and compare it with the original LightGCN++, which does not correct for bias, and APDA, which is the best baseline on the KuaiRec semi-synthetic data. For a fair comparison, APDA is also applied to the LightGCN++ backbone. We show the Recall@20 results in Figure 2 and report results for all three metrics in Table 7 in the Appendix. DPAA outperforms across all bias severities from $s = 1$ to $s = 9$. DPAA performs comparably to LightGCN++ under mild bias, with substantially larger gains as severity increases. This also suggests that DPAA avoids over-correction and under-correction while effectively mitigating popularity bias. These results suggest that, when applied to a different backbone, DPAA remains effective at mitigating popularity bias compared to other methods, especially with high bias severity.

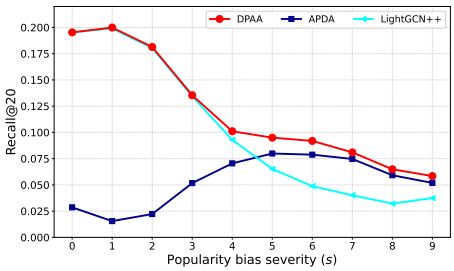


Figure 2: Performance using LightGCN++ as the backbone on the KuaiRec dataset.

Ablation studies. We evaluate the contribution of each DPAA component. Table 9 and Figure 6 in the Appendix show Recall@20 on KuaiRec semi-synthetic click data with varying bias severity. Applying IIW to initial layers generally outperforms applying it to all layers, suggesting that aggressive correction may remove useful higher-order collaborative signals. For example, at bias severity levels 8 and 9, applying IIW only to the initial layer improves performance over applying it to all layers by 8.47% and 4.81%. Removing either the pre-trained or current model estimate degrades performance, while removing IIW or LW generally leads to further performance drops. For instance, at bias severity levels 8 and 9, DPAA improves Recall@20 by 22.01% and 25.84% over the variant without the current model for IIW estimation, by 18.67% and 18.79% over the variant without the pre-trained model, and by 40.28% and 27.07% over the variant without LW, respectively. These results show the importance of combining both models for IIW along with LW. Detailed component analyses are provided in Appendix A.5.1. We also evaluate DPAA using pre-trained embeddings of varying quality (Table 10 in the Appendix). Better-quality pre-trained models generally improve performance, while lower-quality models cause only minor degradation, demonstrating DPAA's robustness to the choice of pre-trained model (Appendix A.5.2). We conduct hyperparameter ablations by varying δ, η, and C. For δ, smaller values perform best on Coat and KuaiRec, with $\delta = 0.2$ yielding the best performance, while $\delta = 0.6$ performs best on Yahoo! R3 (Figure 7). For η, the optimal value varies: $\eta = 2$ for Coat, $\eta = 3$ for KuaiRec, and $\eta = 0$ for sparse Yahoo! R3 (Figure 8). For C, small nonzero values perform best on Coat and KuaiRec, while $C = 0$ performs best on Yahoo! R3 (Figure 9). Detailed analyses are provided in Appendix A.5.3.

5 Conclusion

We propose DPAA, a popularity debiasing method for GNN-based CF that mitigates popularity amplification during message passing. DPAA combines adaptive interaction-level weighting with layer-wise weighting to reduce the influence of popular items during message passing and emphasize higher-order collaborative signals. Our theoretical analyses show that DPAA reduces the relative contribution of popular items while increasing the role of higher-order signals in representation learning. We also introduce a semi-synthetic click generation process to control popularity bias for systematic evaluation. Unbiased evaluations on real-world and semi-synthetic data show that DPAA outperforms existing methods, improves performance for both popular and niche items, and remains effective across the evaluated backbones and bias severities, with larger gains under severe bias.

AI USE STATEMENT

In this work, we did not use generative AI tools for any tasks requiring disclosure. We used generative AI tools (e.g., ChatGPT) to modify scientific figures, edit software code, improve the readability of the research paper, and search for information. We reviewed all AI-assisted work. For example, we used LLMs to modify LaTeX code to format a figure and verified that the resulting figure conveyed what we intended. We also used LLMs to modify small parts of the software code and verified and tested the changes for correctness. We used LLMs to polish our writing for grammar, word choice, and readability, and verified that the revised texts were accurate and conveyed our intended meaning. We used LLMs to search for relevant information and verified the information against its original sources. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

In this work, we introduce a method to mitigate popularity bias in GNN-based CF. To facilitate evaluation and reproducibility, we provide an anonymous repository containing the source code, datasets, and clear instructions for automatically running the method. Anonymous code is available here: `https://anonymous.4open.science/r/DPAA`. We will make the repository public upon acceptance. Moreover, we provide detailed implementation information, parameter settings, hyperparameter values, backbone models, and other relevant documentation needed to reproduce results in Subsection A.3.7 in Appendix.

REFERENCES

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pp. 42–46, 2017.

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. In *FLAIRS*, 2019.

Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. Unbiased learning to rank with unbiased propensity estimation. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 385–394, 2018.

Stephen Bonner and Flavian Vasile. Causal embeddings for recommendation. In *Proceedings of the 12th ACM conference on recommender systems*, pp. 104–112, 2018.

Ludovico Boratto, Gianni Fenu, and Mirko Marras. Connecting user and item perspectives in popularity debiasing for collaborative recommendation. *Information Processing & Management*, 58 (1):102387, 2021.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 14(1):3207–3260, 2013.

Rocío Cañamares and Pablo Castells. Should i follow the crowd? a probabilistic analysis of the effectiveness of popularity in recommender systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 415–424, 2018.

Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM conference on recommender systems*, pp. 224–232, 2018.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3438–3445, 2020a.

Jiajia Chen, Jiancan Wu, Jiawei Chen, Xin Xin, Yong Li, and Xiangnan He. How graph convolutions amplify popularity bias for recommendation? *Frontiers of Computer Science*, 18(5):185603, 2024.

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

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735. PMLR, 2020b.

Zhihong Chen, Rong Xiao, Chenliang Li, Gangfeng Ye, Haochuan Sun, and Hongbo Deng. Esam: Discriminative domain adaptation with non-displayed items to improve long-tail performance. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 579–588, 2020c.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020.

Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. Kuairec: A fully-observed dataset and insights for evaluating recommender systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 540–550, 2022.

Alois Gruson, Praveen Chandar, Christophe Charbuillet, James McInerney, Samantha Hansen, Damien Tardieu, and Ben Carterette. Offline evaluation to make decisions about playlistrecommendation algorithms. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 420–428, 2019.

Ming He, Changshu Li, Xinlei Hu, Xin Chen, and Jiwen Wang. Mitigating popularity bias in recommendation via counterfactual inference. In *International Conference on Database Systems for Advanced Applications*, pp. 377–388. Springer, 2022.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 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*, pp. 639–648, 2020.

Yanbiao Ji, Yue Ding, Dan Luo, Chang Liu, Yuxiang Lu, Xin Xin, and Hongtao Lu. How does topology bias distort message passing in graph recommender? a dirichlet energy perspective. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the tenth ACM international conference on web search and data mining*, pp. 781–789, 2017.

Minseok Kim, Jinoh Oh, Jaeyoung Do, and Sungjin Lee. Debiasing neighbor aggregation for graph neural network in recommender systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 4128–4132, 2022.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *The Fifth International Conference on Learning Representations*, 2017.

Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1748–1757, 2020.

Geon Lee, Kyungho Kim, and Kijung Shin. Post-training embedding enhancement for long-tail recommendation. In *Proceedings of the 33rd ACM international conference on information and knowledge management*, pp. 3857–3861, 2024.

Geon Lee, Kyungho Kim, Fanchen Bu, Langzhang Liang, and Kijung Shin. Revisiting lightgcn: Unexpected inflexibility, inconsistency, and a remedy towards improved recommendation. *ACM Transactions on Recommender Systems*, 4(4):1–26, 2026.

Allen Lin, Jianling Wang, Ziwei Zhu, and James Caverlee. Quantifying and mitigating popularity bias in conversational recommender systems. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pp. 1238–1247, 2022a.

Siyi Lin, Chongming Gao, Jiawei Chen, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. How do recommendation models amplify popularity bias? an analysis from the spectral perspective. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pp. 659–668, 2025.

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

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

Yuanhao Liu, Qi Cao, Huawei Shen, Yunfan Wu, Shuchang Tao, and Xueqi Cheng. Popularity debiasing from exposure to interaction in collaborative filtering. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pp. 1801–1805, 2023.

Sebastian Lubos, Alexander Felfernig, and Markus Tautschnig. An overview of video recommender systems: state-of-the-art and research issues. *Frontiers in big Data*, 6:1281614, 2023.

Dan Luo, Lixin Zou, Qingyao Ai, Zhiyu Chen, Dawei Yin, and Brian D Davison. Model-based unbiased learning to rank. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 895–903, 2023.

Dan Luo, Lixin Zou, Qingyao Ai, Zhiyu Chen, Chenliang Li, Dawei Yin, and Brian D Davison. Unbiased learning-to-rank needs unconfounded propensity estimation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1535–1545, 2024.

Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. Ultragcn: ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pp. 1253–1262, 2021.

Benjamin M Marlin and Richard S Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*, pp. 5–12, 2009.

Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5): 323–351, 2005.

Wentao Ning, Reynold Cheng, Xiao Yan, Ben Kao, Nan Huo, Nur Al Hasan Haldar, and Bo Tang. Debiasing recommendation with personal popularity. In *Proceedings of the ACM Web Conference 2024*, pp. 3400–3409, 2024.

Harrie Oosterhuis and Maarten de Rijke. Policy-aware unbiased learning to rank for top-k rankings. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 489–498, 2020.

Shaowen Peng, Kazunari Sugiyama, and Tsunenori Mine. Svd-gcn: A simplified graph convolution paradigm for recommendation. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pp. 1625–1634, 2022.

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

Yuta Saito. Doubly robust estimator for ranking metrics with post-click conversions. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pp. 92–100, 2020.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pp. 1670–1679, 2016.

Xiaoyu Shi, Quanliang Liu, Hong Xie, Di Wu, Bo Peng, MingSheng Shang, and Defu Lian. Relieving popularity bias in interactive recommendation: A diversity-novelty-aware reinforcement learning approach. *ACM Transactions on Information Systems*, 42(2):1–30, 2023.

Harald Steck. Calibrated recommendations. In *Proceedings of the 12th ACM conference on recommender systems*, pp. 154–162, 2018.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. In *International conference on learning representations*, volume 6, pp. 2. Ithaca, 2018.

Chen Wang, Yueqing Liang, Zhiwei Liu, Tao Zhang, and Philip S Yu. Pre-training graph neural network for cross domain recommendation. In *2021 IEEE Third International Conference on Cognitive Machine Intelligence (CogMI)*, pp. 140–145. IEEE, 2021a.

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

Wenjie Wang, Fuli Feng, Xiangnan He, Xiang Wang, and Tat-Seng Chua. Deconfounded recommendation for alleviating bias amplification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 1717–1725, 2021b.

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

Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. Disentangled graph collaborative filtering. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 1001–1010, 2020a.

Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 115–124, 2016.

Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 610–618, 2018.

Zifeng Wang, Xi Chen, Rui Wen, Shao-Lun Huang, Ercan Kuruoglu, and Yefeng Zheng. Information theoretic counterfactual learning from missing-not-at-random feedback. *Advances in Neural Information Processing Systems*, 33:1854–1864, 2020b.

Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. Model-agnostic counterfactual reasoning for eliminating popularity bias in recommender system. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 1791–1800, 2021.

Hongyi Wen, Xinyang Yi, Tiansheng Yao, Jiaxi Tang, Lichan Hong, and Ed H Chi. Distributionally-robust recommendations for improving worst-case user experience. In *Proceedings of the ACM web conference 2022*, pp. 3606–3610, 2022.

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

Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *Proceedings of the 12th ACM conference on recommender systems*, pp. 279–287, 2018.

Sirui Yao and Bert Huang. Beyond parity: Fairness objectives for collaborative filtering. *Advances in neural information processing systems*, 30, 2017.

Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 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*, pp. 1294–1303, 2022.

Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. Xsimgcl: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 36(2):913–926, 2023.

Bowen Yuan, Yaxu Liu, Jui-Yang Hsia, Zhenhua Dong, and Chih-Jen Lin. Unbiased ad click prediction for position-aware advertising systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pp. 368–377, 2020.

An Zhang, Wenchang Ma, Xiang Wang, and Tat-Seng Chua. Incorporating bias-aware margins into contrastive loss for collaborative filtering. *Advances in Neural Information Processing Systems*, 35:7866–7878, 2022.

An Zhang, Leheng Sheng, Zhibo Cai, Xiang Wang, and Tat-Seng Chua. Empowering collaborative filtering with principled adversarial contrastive loss. *Advances in Neural Information Processing Systems*, 36:6242–6266, 2023a.

Junwei Zhang, Min Gao, Junliang Yu, Lei Guo, Jundong Li, and Hongzhi Yin. Double-scale self-supervised hypergraph learning for group recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pp. 2557–2567, 2021a.

Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. Causal intervention for leveraging popularity bias in recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 11–20, 2021b.

Yin Zhang, Ruoxi Wang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, James Caverlee, and Ed H Chi. Empowering long-tail item recommendation through cross decoupling network (cdn). In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5608–5617, 2023b.

Chu Zhao, Enneng Yang, Yuliang Liang, Pengxiang Lan, Yuting Liu, Jianzhe Zhao, Guibing Guo, and Xingwei Wang. Graph representation learning via causal diffusion for out-of-distribution recommendation. In *Proceedings of the ACM on Web Conference 2025*, pp. 334–346, 2025a.

Chu Zhao, Enneng Yang, Yuliang Liang, Jianzhe Zhao, Guibing Guo, and Xingwei Wang. Distributionally robust graph out-of-distribution recommendation via diffusion model. In *Proceedings of the ACM on Web Conference 2025*, pp. 2018–2031, 2025b.

Zihao Zhao, Jiawei Chen, Sheng Zhou, Xiangnan He, Xuezhi Cao, Fuzheng Zhang, and Wei Wu. Popularity bias is not always evil: Disentangling benign and harmful bias for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(10):9920–9931, 2022.

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

Huachi Zhou, Hao Chen, Junnan Dong, Daochen Zha, Chuang Zhou, and Xiao Huang. Adaptive popularity debiasing aggregator for graph collaborative filtering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 7–17, 2023.

Ziwei Zhu, Yun He, Xing Zhao, and James Caverlee. Popularity bias in dynamic recommendation. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 2439–2449, 2021a.

Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. Popularity-opportunity bias in collaborative filtering. In *Proceedings of the 14th ACM international conference on web search and data mining*, pp. 85–93, 2021b.

A Appendix

A.1 Related Work

Existing approaches for mitigating popularity bias fall into several categories: re-weighting or IPW, causal inference, regularization-based methods, and post-processing methods. Re-weighting approaches, commonly known as IPW methods (Ai et al., 2018; Gruson et al., 2019; Joachims et al., 2017; Saito, 2020; Schnabel et al., 2016; Yang et al., 2018), treat item popularity as a proxy for exposure probability and correct bias by weighting each instance inversely to that probability. To mitigate the high variance problem with such re-weighting, several studies (Bottou et al., 2013; Gruson et al., 2019) incorporate additional strategies, such as normalization techniques or smoothing regularization, to stabilize the training dynamics and produce more reliable estimates (Bottou et al., 2013; Gruson et al., 2019). IPW has also been extensively adopted in learning-to-rank systems to address different biases, including position, selection, and popularity bias (Joachims et al., 2017; Oosterhuis & de Rijke, 2020; Wang et al., 2016; Ai et al., 2018; Luo et al., 2024; 2023; Saito, 2020; Yuan et al., 2020; Wang et al., 2018).

Regularization-based approaches (Abdollahpouri et al., 2017; Boratto et al., 2021; Chen et al., 2020c; Lin et al., 2025; Liu et al., 2023; Zhu et al., 2021b) address popularity bias by adding penalty terms that balance accuracy and coverage in recommendations. These methods differ in their penalty designs: Reg (Zhu et al., 2021b) separates item popularity from predicted user preferences; ESAM (Chen et al., 2020c) employs attribute alignment, clustering, and self-training regularization to improve feature representations; ALS+Reg (Abdollahpouri et al., 2017) promotes fairness via intra-list diversity penalties; sam-reg (Boratto et al., 2021) reduces the biased correlation between relevance and item popularity, IPL applies a fairness criterion via a regularization term that ensures interactions are proportional to the number of users, and ReSN (Lin et al., 2025) applies a spectral norm regularizer to penalize the popularity-induced principal singular value. Causal inference has been used to separate user interests from popularity effects (Bonner & Vasile, 2018; He et al., 2022; Ning et al., 2024; Wang et al., 2021b; Wei et al., 2021; Zhang et al., 2021b; Zhao et al., 2022; Zheng et al., 2021) by leveraging counterfactual reasoning to identify and reduce the impact of popularity on recommendations. CausE (Bonner & Vasile, 2018) employs causal graphs for unbiased learning, MACR (Wei et al., 2021) removes direct popularity effects through multi-task counterfactual learning, MPCI (He et al., 2022) blocks paths from popularity to predictions, and PPAC (Ning et al., 2024) jointly models global and personal popularity within user neighborhoods.

Post-processing re-ranking methods (Abdollahpouri et al., 2019; Ji et al., 2025; Steck, 2018; Zhu et al., 2021a;b) typically operate on the generated recommendations without altering model parameters. Their objectives differ depending on the method: Calibration (Steck, 2018) preserves the historical preference distribution of users, RankALS (Abdollahpouri et al., 2019) enhances recommendation diversity, and FPC (Zhu et al., 2021a) addresses popularity bias in dynamic settings by adjusting predicted scores. Some studies have also investigated debiasing through out-of-distribution (OOD) generalization (Wen et al., 2022; Zhao et al., 2025a;b), embedding landscape modeling (Lee et al., 2024), and reinforcement learning approaches (Lin et al., 2022a; Shi et al., 2023), with the goal of mitigating bias while maintaining a balance between recommendation accuracy and fairness. Moreover, CDN (Zhang et al., 2023b) disentangles memorization from user behavior to alleviate long-tail bias in content-based recommendation systems.

GNNs (Defferrard et al., 2016; Veličković et al., 2018) have become a widely used method for capturing complex user-item interactions in recommender systems. Early models, such as NGCF (Wang et al., 2019), employ iterative graph convolution to propagate collaborative signals, while more recent architectures, such as LightGCN (He et al., 2020), UltraGCN (Mao et al., 2021), and SVD-GCN (Peng et al., 2022), focus on enhancing computational efficiency and prediction accuracy. LightGCN (He et al., 2020) removes feature transformations and non-linear activations to retain only linear neighborhood aggregation, UltraGCN (Mao et al., 2021) bypasses explicit message passing entirely using a constraint-based objective, and SVD-GCN (Peng et al., 2022) approximates graph convolution via truncated singular value decomposition for more efficient training. More recently, LightGCN++ (Lee et al., 2026) extends LightGCN with refined neighborhood aggregation and layer-wise representation combination to improve embedding quality and recommendation performance.

Incorporating contrastive and self-supervised learning (Wang et al., 2021a; Wu et al., 2021; Yu et al., 2023; 2022; Zhang et al., 2021a), as in SimGCL (Yu et al., 2022) and XSimGCL (Yu et al., 2023) has

further strengthened representation quality through graph augmentations. DirectAU (Wang et al., 2022) emphasizes alignment and uniformity in representations, while SGL (Wu et al., 2021) and NCL (Lin et al., 2022b) strengthen robustness by integrating contrastive learning with structural and semantic signals. GCF$_{logdet}$ (Zhang et al., 2023a) adds a regularization penalty during training to keep user and item embeddings diverse and spread out, preventing them from collapsing around popular items. Despite their effectiveness, these methods do not address the popularity amplification inherent in GNNs. APDA (Zhou et al., 2023) mitigates this by applying inverse popularity weights to interactions during aggregation, and NAVIP (Kim et al., 2022) leverages IPW based on static interaction counts during aggregation. DAP (Chen et al., 2024) is a post-hoc GNN-based debiasing method that estimates item popularity by clustering nodes with similar characteristics to remove the popularity component from embeddings. TSP (Ji et al., 2025) mitigates topology-induced bias to improve long-tail item recommendations. Because TSP only applies corrections during inference, it cannot fully address bias that was baked into the embeddings during training. In contrast, our method directly intervenes during message passing, the source of bias amplification in GNNs, adjusting popularity-biased signals at the aggregation level.

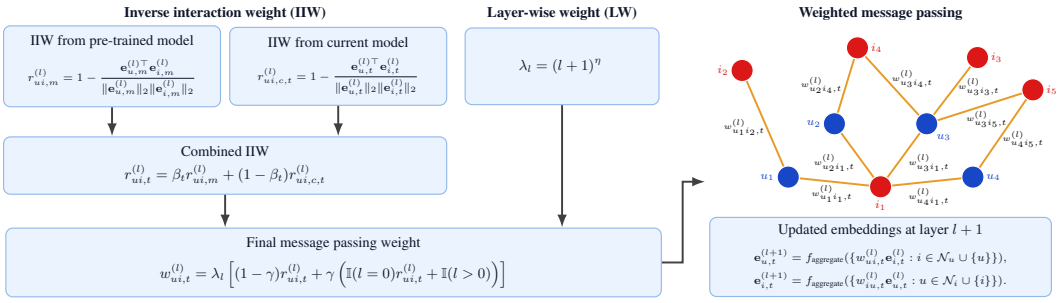


Figure 3: Illustration of the weighted message passing mechanism of DPAA. IIW estimates from the pre-trained and current models are combined for each user–item pair at epoch t, then combined with LW computed from the layer index to obtain final interaction weights. These weights are used in neighborhood aggregation to produce updated embeddings at layer $(l+1)$. Blue nodes denote users, red nodes denote items, and each interaction is annotated with its message passing weight.

A.2 Theoretical Analysis

A.2.1 Proof of Lemma 3.1

In the main body of the paper, we define the message-passing contributions as $\mathcal{I}_i^{(l)} = d_i \cdot \|\mathbf{e}_{i,t}^{(l)}\|_2$ and $\mathcal{I}_{\mathrm{IIW}_i}^{(l)} = d_i \cdot \bar{r}_{i,t}^{(l)} \cdot \|\mathbf{e}_{i,t}^{(l)}\|_2$, where $\bar{r}_{i,t}^{(l)} = \frac{1}{d_i}\sum_{u\in\mathcal{N}(i)} r_{ui,t}^{(l)}$ is the average IIW of item i.

In standard GNN aggregation, the ratio of the message-passing contributions of a popular item p and a long-tail item q is:

$$\frac{\mathcal{I}_p^{(l)}}{\mathcal{I}_q^{(l)}} = \frac{d_p \,\|\mathbf{e}_{p,t}^{(l)}\|_2}{d_q \,\|\mathbf{e}_{q,t}^{(l)}\|_2},$$

which is strictly positive since the degrees and embedding norms are positive. Under IIW, this ratio becomes:

$$\frac{\mathcal{I}_{\mathrm{IIW}_p}^{(l)}}{\mathcal{I}_{\mathrm{IIW}_q}^{(l)}} = \frac{d_p\,\bar{r}_{p,t}^{(l)}\,\|\mathbf{e}_{p,t}^{(l)}\|_2}{d_q\,\bar{r}_{q,t}^{(l)}\,\|\mathbf{e}_{q,t}^{(l)}\|_2} = \frac{\bar{r}_{p,t}^{(l)}}{\bar{r}_{q,t}^{(l)}} \cdot \frac{\mathcal{I}_p^{(l)}}{\mathcal{I}_q^{(l)}}.$$

Subtracting the IIW-weighted ratio from the standard ratio gives:

$$\begin{aligned}\frac{\mathcal{I}_p^{(l)}}{\mathcal{I}_q^{(l)}} - \frac{\mathcal{I}_{\mathrm{IIW}_p}^{(l)}}{\mathcal{I}_{\mathrm{IIW}_q}^{(l)}} &= \frac{\mathcal{I}_p^{(l)}}{\mathcal{I}_q^{(l)}} - \frac{\bar{r}_{p,t}^{(l)}}{\bar{r}_{q,t}^{(l)}} \cdot \frac{\mathcal{I}_p^{(l)}}{\mathcal{I}_q^{(l)}} \\ &= \left(1 - \frac{\bar{r}_{p,t}^{(l)}}{\bar{r}_{q,t}^{(l)}}\right) \cdot \frac{\mathcal{I}_p^{(l)}}{\mathcal{I}_q^{(l)}}.\end{aligned}$$

By the user-item similarity property, motivated by prior work (Zhou et al., 2023), popular items are more broadly similar with users than long-tail items. Thus, $\bar{r}_{p,t}^{(l)} < \bar{r}_{q,t}^{(l)}$ with $\bar{r}_{q,t}^{(l)} > 0$, which implies $1 - \bar{r}_{p,t}^{(l)}/\bar{r}_{q,t}^{(l)} > 0$. Combined with the positivity of $\mathcal{I}_p^{(l)}/\mathcal{I}_q^{(l)}$, the product is strictly positive, establishing both the equality and the positivity in Lemma 3.1.

Lemma 3.1 has two implications for understanding IIW. First, the larger the popularity gap between popular and long-tail items, reflected by a smaller IIW ratio $\bar{r}_{p,t}^{(l)}/\bar{r}_{q,t}^{(l)}$, the larger the IIW-induced reduction in popular-item dominance. Second, the larger the structural degree imbalance d_p/d_q, the larger the reduction in the relative contribution ratio, because the standard contribution ratio $\mathcal{I}_p^{(l)}/\mathcal{I}_q^{(l)}$ scales linearly with the degree ratio. Therefore, IIW corrects more strongly when popularity bias is more pronounced, either in the learned embeddings or in the graph topology.

A.2.2 Proof of Lemma 3.2

We analyze how the LW component of DPAA shapes representation learning. To understand the effect of λ_ℓ on the embeddings, we analyze how it affects the gradient of the loss function during backpropagation. The BPR loss is the standard training objective in LightGCN, and the gradient of this loss with respect to a node's embedding directly determines how that embedding evolves during training. By tracing how λ_ℓ propagates through this gradient, we can understand how LW reshapes the contribution of different layers to representation updates. We perform this analysis using a simplified LightGCN-style propagation for simplicity. For notational simplicity, we omit the LightGCN degree-normalization constants since including them does not change the role of λ_ℓ in the gradient expression.

For a training triple (u, i, j), where u is a user, i is an item with which u has interacted, and j is a randomly sampled item with which u has not interacted, the BPR loss is:

$$\mathcal{L}_{uij} = -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}),$$

where $\hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i$, $\hat{y}_{uj} = \mathbf{e}_u^\top \mathbf{e}_j$, and $\mathbf{e}_u$, $\mathbf{e}_i$, and $\mathbf{e}_j$ are the final embeddings.

Let $s_{uij} = \hat{y}_{ui} - \hat{y}_{uj} = \mathbf{e}_u^\top(\mathbf{e}_i - \mathbf{e}_j)$. Differentiating the BPR loss with respect to s_{uij} gives:

$$\frac{\partial \mathcal{L}_{uij}}{\partial s_{uij}} = -\epsilon_{uij}, \qquad \epsilon_{uij} = 1 - \sigma(s_{uij}).$$

By the chain rule, for any node $v \in \{u, i, j\}$:

$$\frac{\partial \mathcal{L}_{uij}}{\partial \mathbf{e}_v} = -\epsilon_{uij} \cdot \frac{\partial s_{uij}}{\partial \mathbf{e}_v}. \tag{11}$$

The gradients of s_{uij} with respect to node embeddings are:

$$\frac{\partial s_{uij}}{\partial \mathbf{e}_u} = \mathbf{e}_i - \mathbf{e}_j, \qquad \frac{\partial s_{uij}}{\partial \mathbf{e}_i} = \mathbf{e}_u, \qquad \frac{\partial s_{uij}}{\partial \mathbf{e}_j} = -\mathbf{e}_u. \tag{12}$$

For the gradient analysis, we first express the final embedding of a node v using the LightGCN readout function in terms of its layer-wise representations as:

$$\mathbf{e}_v = \frac{1}{L+1} \sum_{\ell=0}^{L} \mathbf{e}_v^{(\ell)}.$$

Thus, for item i:

$$\mathbf{e}_i = \frac{1}{L+1} \sum_{\ell=0}^{L} \mathbf{e}_i^{(\ell)}.$$

Substituting the LightGCN-style message passing with LW: $\mathbf{e}_i^{(\ell)} = \lambda_{\ell-1} \sum_{u' \in \mathcal{N}(i)} \mathbf{e}_{u'}^{(\ell-1)}$, using $\lambda_0 = 1$ and $\lambda_{\ell-1} = \ell^\eta$, we obtain:

$$\mathbf{e}_i = \frac{1}{L+1} \left(\mathbf{e}_i^{(0)} + \sum_{\ell=1}^{L} \ell^\eta \sum_{u' \in \mathcal{N}(i)} \mathbf{e}_{u'}^{(\ell-1)} \right).$$

The same expansion applies to $\mathbf{e}_j$ and $\mathbf{e}_u$:

$$\mathbf{e}_j = \frac{1}{L+1}\left(\mathbf{e}_j^{(0)} + \sum_{\ell=1}^{L} \ell^{\eta} \sum_{u' \in \mathcal{N}(j)} \mathbf{e}_{u'}^{(\ell-1)}\right),$$

and

$$\mathbf{e}_u = \frac{1}{L+1}\left(\mathbf{e}_u^{(0)} + \sum_{\ell=1}^{L} \ell^{\eta} \sum_{i' \in \mathcal{N}(u)} \mathbf{e}_{i'}^{(\ell-1)}\right).$$

Substituting these expansions into equation 11, using values from equation 12, and combining the three cases gives the gradient with respect to any node $v \in \{u, i, j\}$:

$$\frac{\partial \mathcal{L}_{uij}}{\partial \mathbf{e}_v} = \begin{cases} \frac{-\epsilon_{uij}}{L+1}\left(\mathbf{e}_i^{(0)} - \mathbf{e}_j^{(0)} + \sum_{\ell=1}^{L} \ell^{\eta}\left[\sum_{u' \in \mathcal{N}(i)} \mathbf{e}_{u'}^{(\ell-1)} - \sum_{u' \in \mathcal{N}(j)} \mathbf{e}_{u'}^{(\ell-1)}\right]\right), & \text{if } v = u, \\ \frac{-\epsilon_{uij}}{L+1}\left(\mathbf{e}_u^{(0)} + \sum_{\ell=1}^{L} \ell^{\eta} \sum_{i' \in \mathcal{N}(u)} \mathbf{e}_{i'}^{(\ell-1)}\right), & \text{if } v = i, \\ \frac{\epsilon_{uij}}{L+1}\left(\mathbf{e}_u^{(0)} + \sum_{\ell=1}^{L} \ell^{\eta} \sum_{i' \in \mathcal{N}(u)} \mathbf{e}_{i'}^{(\ell-1)}\right), & \text{if } v = j. \end{cases}$$

In each case, since $\frac{\epsilon_{uij}}{L+1}$ is a scalar constant with respect to the embeddings, the gradient is proportional to the remaining term, which can be written as:

$$\frac{\partial \mathcal{L}_{uij}}{\partial \mathbf{e}_v} \propto \phi_v(\mathbf{e}^{(0)}) + \sum_{\ell=1}^{L} \ell^{\eta} \cdot \psi_{\ell}(\mathbf{h}_v^{(\ell)})$$

where $\phi_v(\mathbf{e}^{(0)})$ is a function of the initial embeddings, and $\psi_{\ell}(\mathbf{h}_v^{(\ell)})$ is a function of the layer-ℓ neighborhood aggregation of node v. In each gradient expression, the contribution from a layer embeddings carries a multiplicative factor of ℓ^{η}. The downstream effect on representation learning is the following.

In each gradient expression, the contribution from a layer-ℓ embedding carries a multiplicative factor of ℓ^{η}, which directly determines how strongly that layer participates in shaping the representation update. The downstream effect on representation learning can be characterized along two complementary directions.

First, structural amplification of deeper layers. For any two layers ℓ_a and ℓ_b with $\ell_a < \ell_b$, the deeper layer carries a strictly larger gradient coefficient $(\ell_a)^{\eta} < (\ell_b)^{\eta}$ whenever $\eta > 0$. In contrast, a standard GNN without LW corresponds to the special case $\eta = 0$, where $\ell^{\eta} = 1$ for every layer and all layers contribute equally to the gradient, regardless of their depth. By setting $\eta > 0$, LW therefore structurally amplifies the gradient influence of deeper layers during backpropagation, reducing the relative contribution of shallow layers whose updates are driven primarily by immediate user–item interactions. Since deeper layers aggregate signals from multi-hop neighborhoods, this re-weighting distributes representation learning across a broader set of items reachable through indirect collaborative paths, including diverse and potentially long-tail items that lie beyond a user's immediate neighborhood.

Second, monotonic strengthening with η. As η increases, the gradient-coefficient ratio $(\ell_b)^{\eta}/(\ell_a)^{\eta} > 1$ between any deeper layer ℓ_b and any shallower layer ℓ_a grows monotonically, progressively amplifying the contribution of deeper layers to embedding updates. This provides a mechanism for controlling how strongly higher-order collaborative signals influence representation learning relative to shallow ones. In scenarios where immediate interactions are heavily dominated by popular items, as is typical under severe popularity skew, a larger η can be particularly beneficial, since it shifts gradient influence away from popularity-skewed local interactions and toward higher-order collaborative signals carried through multi-hop neighborhoods. Conversely, when popularity bias is mild, smaller values of η can suffice, as shallow-layer updates are not as heavily distorted by popularity-driven interactions.

Together, these two implications from Lemma 3.2 explain how LW reshapes representation learning: it not only amplifies gradient influence toward deeper layers in a structural sense, but also provides

a tunable knob (η) that scales this amplification with the severity of local dominance of popularity bias in the data, consistent with the motivation for incorporating LW into message passing.

A.3 Additional Experimental Details

A.3.1 Dataset details

We use three real-world datasets with unbiased test sets: Coat (Schnabel et al., 2016), KuaiRec (Gao et al., 2022), and Yahoo! R3 (Marlin & Zemel, 2009). Coat contains user ratings from an online coat shopping platform. Coat contains 290 users, 300 items, and 5,490 interactions, with a sparsity of 0.06310. In the training set, each user rates 24 self-selected items, while the test set is constructed by randomly exposing 16 items to each user. Ratings of 3 or higher on the 5-point scale are treated as positive feedback. KuaiRec is derived from Kuaishou short-video logs and provides a fully observed user–item matrix for evaluation, where nearly all users interact with almost every video. KuaiRec contains 7,176 users, 10,728 items, and 1,153,106 interactions overall, with a sparsity of 0.01498. Although this test setting is dense, the training data remains sparse. Its test set contains ratings from 1,411 users over 3,327 items. Following (Gao et al., 2022), we treat an interaction as positive if the watch time exceeds twice the video duration. Yahoo! R3 includes user–song ratings collected under self-selection. Yahoo! R3 contains 14,382 users, 1,000 items, and 129,748 interactions, with a sparsity of 0.00902. Yahoo! R3 is substantially sparser than Coat and KuaiRec, allowing us to evaluate model performance under a more challenging sparse-feedback setting. Its unbiased test set asks 5,400 users to rate 10 randomly selected songs. Ratings of 3 or higher are treated as positive feedback. We select these datasets because their unbiased test sets enable unbiased evaluation of recommendation methods under real-world settings. Moreover, their differences in scale, sparsity, and popularity distribution allow us to assess model performance across diverse recommendation scenarios. Dataset statistics are summarized in Table 3.

Table 3: Dataset statistics.

Dataset	#Users	#Items	#Interactions	Sparsity
Coat	290	300	5,490	0.06310
KuaiRec	7,176	10,728	1,153,106	0.01498
Yahoo! R3	14,382	1,000	129,748	0.00902

A.3.2 Popularity Distribution of the Datasets

To examine the popularity imbalance in the datasets used in our experiments, we analyze how training interactions are distributed across popular and niche items. This analysis illustrates the extent to which interactions are concentrated among popular items in each dataset. Figure 4 shows the training data distributions for the three real-world datasets used in our experiments. For each dataset, popular items are defined as the smallest set of items that accounts for 80% of observed training interactions, while the remaining items are treated as niche or long-tail items and account for the remaining 20%. On Coat, 54.5% of items account for 80% of training interactions, compared with only 21.1% on KuaiRec and 34.8% on Yahoo! R3. These proportions reveal differences in the degree of popularity imbalance across the datasets. KuaiRec exhibits the strongest popularity skew, with most interactions concentrated among a relatively small fraction of items. Yahoo! R3 shows moderate skew, while Coat exhibits the mildest skew among the three datasets, with interactions spread across a larger proportion of items. Overall, these results show that interaction distributions in these real-world datasets are skewed toward popular items, although the extent of this skew varies across datasets.

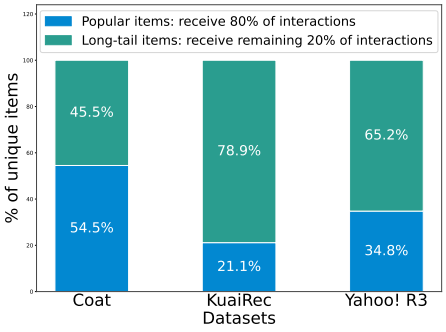


Figure 4: Training data distribution across real-world datasets. Popular items account for 80% of interactions, while the remaining items form the long-tail group.

A.3.3 RELATIONSHIP BETWEEN USER–ITEM SIMILARITY AND ITEM POPULARITY

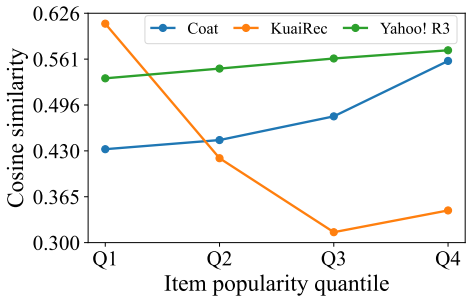


Figure 5: Average user–item cosine similarity across item popularity quartiles. Items are divided into four equally sized groups based on interaction frequency: Q1 contains the least popular 25% of items, Q2 the 26–50% range, Q3 the 51–75% range, and Q4 the rest most popular 25% items.

Table 4: Percentage of training interactions across item popularity quartiles. Each value indicates the share of training interactions associated with each item quartile.

Quartile	Coat	KuaiRec	Yahoo! R3
Q1 (1-25%)	7.4%	0.4%	4.0%
Q2 (26-50%)	15.8%	2.0%	8.2%
Q3 (51-75%)	24.5%	12.3%	14.7%
Q4 (76-100%)	52.3%	85.3%	73.0%

We empirically examine the relationship between item popularity and user–item similarity, which motivates our interaction-level weighting mechanism, where interactions with higher user–item similarity are more likely to involve popular items. We rank items by their training interaction frequency and divide them into four equally sized popularity quartiles: Q1 contains the bottom 25% (least popular items), Q2 the 25–50% range, Q3 the 50–75% range, and Q4 the 75–100% range (most popular 25%). For each quartile, we compute the average cosine similarity over observed user–item interactions using the pre-trained model, as shown in Figure5, and also report the percentage of training interactions belonging to each quartile alongside it in Table 4. Coat dataset exhibits a clear increasing trend, with the average cosine similarity increasing from Q1 to Q4. Yahoo! R3 shows a similar overall pattern, with higher average similarity for more popular items. KuaiRec shows a different distribution: the first two popularity quartiles (Q1 and Q2) account for only 2.4% of the observed interactions, whereas Q3 and Q4 together account for approximately 97.6%. Although Q1 and Q2 show higher similarity than Q3, they account for only a small fraction of interactions, and therefore have little influence on the overall interaction distribution. Thus, Q3 and Q4 are representative of the interaction distribution in KuaiRec. Within these two dominant groups, the average cosine similarity increases from Q3 to Q4 as item popularity increases. Overall, these results provide empirical evidence that interactions involving more popular items generally tend to exhibit higher user–item similarity. This supports our use of the complement of cosine similarity as an inverse interaction weight during aggregation, assigning smaller weights to interactions involving popular items and larger weights to those involving niche items.

A.3.4 UNBIASED EVALUATIONS AND RANKING PROTOCOL

We choose the above mentioned three real-world datasets because they provide unbiased test set for unbiased evaluations. While other datasets could also be used for experiments, they typically lack unbiased test sets, as their test data is randomly sampled from observed interactions in the training data (Wei et al., 2021; Zhou et al., 2023). Even when randomly sampled, such test data can still reflect the underlying popularity bias present in the observed interactions. This is because the samples are drawn from popularity-dominated distributions, which limits their suitability for unbiased evaluation. In contrast, the chosen datasets offer unbiased test sets via random exposure or full observation, and span diverse sizes, sparsity levels, and popularity distributions in the training data, reflecting different real-world scenarios.

We follow the all-ranking protocol (Krichene & Rendle, 2020), ranking all candidate items for each user after removing positive training interactions. We report Recall@k, NDCG@k, and Hit Ratio (HR@k), with $k = 20$ by default. For KuaiRec, we follow the modified protocol as in (Zhang et al., 2022), since its test set comes from a densely observed sub-matrix rather than a random sample over the full item catalog (Gao et al., 2022). Thus, following (Zhang et al., 2022), ranking is restricted to the 3,327 fully exposed items in the test sub-matrix.

A.3.5 DETAILS OF SEMI-SYNTHETIC CLICK DATA GENERATION PROCESS

To systematically evaluate how debiasing methods perform under different levels of popularity bias, we construct semi-synthetic training data with controllable bias severity for RQ2. Real-world recommendation datasets typically do not provide direct control over the degree of popularity bias in the training data, making it difficult to analyze method robustness under varying bias intensities. To address this limitation, we generate user–item interactions using a Zipfian probability sampling mechanism Newman (2005) based on item interaction frequencies, which allows us to continuously control the skewness of the training distribution.

We use the original unbiased KuaiRec test set as the basis for dataset construction because it is collected under full-exposure settings and contains many interactions per user. We split this data into three disjoint subsets: 10% for validation, 20% for testing, and the remaining 70% as an unbiased interaction pool for generating popularity-skewed training data. This design ensures that popularity bias is introduced only in the training set, while validation and test sets remain unbiased for fair evaluation.

Before applying probabilistic sampling, we enforce a minimum exposure constraint to avoid degenerate cases. Specifically, for each item, we randomly select one user from the set of users who originally interacted with that item and add the corresponding interaction to the training data, ensuring that every item appears at least once. We then compute a global popularity ranking by counting item frequencies and sorting items in descending order. Let r denote the popularity rank of an item and N denote the total number of items. The Zipfian sampling probability for an item with rank r is defined as:

$$P_{\text{zipf}}(r) = \frac{1/r^s}{\sum_{n=1}^{N} 1/n^s},$$

where s controls the skewness of the distribution. Smaller values of s produce near-uniform sampling, while larger values concentrate more probability on top-ranked popular items. We therefore refer to s as the popularity bias severity parameter. For each user, we normalize these Zipfian probabilities over the user's candidate item set and sample interactions accordingly. This preserves each user's original item pool while injecting global popularity bias into the generated training interactions. By varying s from 0 to 9, we obtain a set of datasets ranging from weakly biased to highly popularity-skewed, enabling systematic evaluation of debiasing methods under increasing popularity bias.

A.3.6 BASELINE DETAILS

Our setup is based on pure CF setting. We use only user-item interactions without any auxiliary item features or user attributes. Therefore, we compare DPAA against baselines that are compatible with this setting. Detailed descriptions of baselines are provided below.

- **APDA** (Zhou et al., 2023) mitigates aggregation-stage bias in GNN-based CF by assigning each edge an adaptive weight derived from an inverse popularity score during message passing.
- **NAVIP** (Kim et al., 2022) debiases neighborhood aggregation in GNNs by using inverse propensity scores as edge weights, reducing the influence of popular items during message passing.
- **DAP** (Chen et al., 2024) is a post-hoc method that clusters structurally similar nodes, estimates the popularity-amplification component of their embeddings, and subtracts it to obtain unbiased representations.
- **PPAC** (Ning et al., 2024) jointly models personal and global popularity within a counterfactual inference framework to attribute and remove popularity-driven prediction effects.
- **MACR** (Wei et al., 2021) applies counterfactual inference within a multi-task learning framework to isolate and subtract the direct causal effect of item popularity on the predicted score.
- **CVIB** (Wang et al., 2020b) applies an information-theoretic counterfactual method to learn unbiased recommendations from missing-not-at-random data by minimizing mutual information to correct for bias.
- **IPS** (Gruson et al., 2019) extends the standard IPW estimator (Joachims et al., 2017) with max-capping and normalization of the propensity weights, which reduces estimator variance.
- **SAM-REG** (Boratto et al., 2021) pairs balanced sampling across popularity tiers with a regularizer that penalizes the correlation between predicted relevance and item popularity.
- **LightGCN** (He et al., 2020) is a GNN-based CF backbone without bias correction.

A.3.7 IMPLEMENTATION DETAILS

Unless stated otherwise, we use LightGCN (He et al., 2020) as the backbone for all baselines and our method, since it is one of the most widely used architectures for GNN-based CF (Zhou et al., 2023). We use the LightGCN implementation of PyTorch and train it with the Adam optimizer. We apply early stopping with a maximum of 1000 epochs and a patience of 50 epochs. Training is terminated if Recall@20 on the validation set does not improve for 50 consecutive epochs, ensuring good performance under the early stopping criterion. We tune the learning rate over $1e^{-5}, 1e^{-4}, 1e^{-3}$. We set the batch size to 2,048 and the embedding dimension to 256, while keeping all other parameters at the default settings of LightGCN. We use a two-layer GCN for the Coat dataset and Yahoo! R3 dataset, and a four-layer GCN for the KuaiRec dataset. Some initial analysis on LightGCN shows that deeper architectures perform better on KuaiRec, whereas fewer layers yield better results on Coat and Yahoo! R3. We therefore apply this same configuration consistently across our method and all baselines for fair comparison. We tune the epoch-dependent coefficient for transitioning from pre-trained to current embeddings over $C \in \{0.0, 1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1e^{0}\}$. We tune the layer-wise weighting parameter, which controls the relative importance of different layers, over $\eta \in \{0, 1, 2, 3\}$. We also tune the initial residual connection parameter δ, which balances initial embeddings and neighborhood aggregation, over the range $[0.0, 1.0]$ with a step size of 0.2. Unless stated otherwise, for the pre-trained model used to estimate IIW, we use a basic LightGCN model without any bias correction.

For RQ4 only, we use the more recent LightGCN++ (Lee et al., 2026) as the backbone to evaluate the generalizability of DPAA to a different GNN-based CF architecture and its ability to mitigate popularity bias in that backbone. In this setting, we also use LightGCN++ as the pre-trained model for estimating IIW. LightGCN++ extends LightGCN by introducing flexible embedding-norm scaling and neighbor weighting during aggregation, together with an adjustable layer-wise embedding pooling mechanism to address the inflexibility and inconsistency of the original LightGCN architecture (Lee et al., 2026). For LightGCN++, we use the implementation provided by the authors and follow the parameter settings of the original LightGCN++ implementation.

Unless stated otherwise, in equation 7, we set $\gamma = 1$ for all experiments, meaning that IIW is applied only at the initial layer. In our ablation studies, we also provide an analysis using $\gamma = 0$, meaning that IIW is applied at all layers in that case. We conduct our experiments on machines with 252 GB RAM, 3 TB storage, a CUDA-enabled GPU with 24 GB memory, 128 CPU cores, and a clock speed of 1.46 GHz.

A.4 ADDITIONAL EXPERIMENTAL RESULTS

A.4.1 MORE ANALYSIS ON RQ1

We now analyze the different parameters of DPAA that produce the best results across datasets and discuss the role of each parameter in adapting our method to different data characteristics. Table 5 summarizes the best parameter combinations of DPAA across datasets: C, which controls the transition from pre-trained to current model embeddings, η, which controls layer-wise weighting, and δ, which controls the initial residual connection.

On Coat, the best parameter combination uses $C = 1e^{-5}$, $\eta = 2.0$, and $\delta = 0.2$. The small but nonzero value of C allows a gradual transition from pre-trained embeddings to current model embeddings, suggesting that current embeddings can also provide useful adaptive signals once training becomes stable. The moderate value of η indicates that higher-order collaborative signals are useful even under milder popularity bias. The small value of δ suggests that preserving a limited amount of initial embedding information is sufficient to reduce over-smoothing while still allowing neighborhood aggregation to shape the representations.

On KuaiRec, the best combination uses $C = 1e^{-3}$, $\eta = 3.0$, and $\delta = 0.2$. The comparatively moderate value of C allows the model to rely on both pre-trained and current model embeddings during training. The larger η is also consistent with the stronger popularity skew in KuaiRec, where emphasizing deeper-layer collaborative signals helps reduce the dominance of shallow popularity-driven interactions, consistent with Lemma 3.2. The small value of δ suggests that preserving a limited amount of initial embedding information is sufficient to reduce over-smoothing.

On Yahoo! R3, the best combination uses $C = 0.0$, $\eta = 0.0$, and $\delta = 0.6$. Since $C = 0.0$ means the model does not use current model embeddings for IIW estimation and relies only on the pre-trained embedding-based weights, this suggests that current embeddings may be unreliable in sparse scenarios, showing the importance of using the pre-trained model. The value $\eta = 0.0$ indicates that uniform layer contribution works better than emphasizing deeper layers. The moderate value of δ suggests that preserving some initial embedding information is necessary while still allowing the model to incorporate higher-order neighbors.

Table 5: Best parameter combinations of DPAA across datasets. Here, C controls the transition from pre-trained to current model embeddings, η controls layer-wise weighting, and δ controls the initial residual connection.

Dataset	C	η	δ
Coat	$1e^{-5}$	2.0	0.2
KuaiRec	$1e^{-3}$	3.0	0.2
Yahoo! R3	0.0	0.0	0.6

Overall, these dataset-specific combinations show that DPAA adapts to different recommendation settings and highlight the importance of its key components, including the combination of pre-trained and current model estimates, the gradual transition between them for IIW estimation, layer-wise weighting, and the preservation of initial embedding information to reduce over-smoothing.

A.4.2 More analysis on RQ2

In the main body of the paper, we present the Recall@20 results under varying popularity bias severity in Figure 1 on the KuaiRec semi-synthetic dataset. For better interpretation of the exact value under different severity levels s, we also report the corresponding Recall@20 results in Table 6.

Table 6: Performance comparison of different methods for Recall@20 on the KuaiRec dataset by varying levels of popularity bias severity (s). Bold and underline indicate the best and second-best results. Results are reported as mean $\pm$ std%. Improvement (%) denotes the relative gain of DPAA over the best baseline.

s	DPAA	APDA	NAVIP	DAP	PPAC	MACR	CVIB	IPS	SAM-REG	LightGCN	Improvement (%)
0.0	**0.2047**$_{\pm 0.70}$	0.1740$_{\pm 0.46}$	0.1557$_{\pm 0.18}$	0.1738$_{\pm 0.83}$	<u>0.2025</u>$_{\pm 0.31}$	0.1213$_{\pm 0.57}$	0.1127$_{\pm 0.09}$	0.0987$_{\pm 0.74}$	0.0956$_{\pm 0.26}$	0.2024$_{\pm 0.91}$	+1.09
1.0	<u>0.2043</u>$_{\pm 1.11}$	0.2001$_{\pm 0.72}$	0.1941$_{\pm 0.34}$	**0.2049**$_{\pm 0.96}$	0.1949$_{\pm 1.12}$	0.0989$_{\pm 1.21}$	0.1762$_{\pm 1.24}$	0.1691$_{\pm 0.48}$	0.1597$_{\pm 0.87}$	0.2001$_{\pm 0.29}$	-0.29
2.0	**0.1859**$_{\pm 1.01}$	0.1763$_{\pm 0.58}$	0.1613$_{\pm 0.75}$	0.1769$_{\pm 0.92}$	0.1810$_{\pm 0.43}$	0.1186$_{\pm 0.76}$	0.1663$_{\pm 0.21}$	0.1210$_{\pm 0.64}$	0.1722$_{\pm 0.35}$	<u>0.1824</u>$_{\pm 0.88}$	+1.92
3.0	**0.1703**$_{\pm 0.53}$	0.1453$_{\pm 0.29}$	0.0920$_{\pm 0.67}$	0.1325$_{\pm 0.41}$	0.1550$_{\pm 0.82}$	0.0944$_{\pm 0.55}$	0.1373$_{\pm 0.32}$	0.0509$_{\pm 1.14}$	0.1600$_{\pm 0.24}$	<u>0.1656</u>$_{\pm 0.61}$	+2.84
4.0	**0.1500**$_{\pm 1.22}$	0.1146$_{\pm 0.81}$	0.0574$_{\pm 1.21}$	0.0798$_{\pm 1.06}$	0.1017$_{\pm 0.52}$	0.0037$_{\pm 0.74}$	0.0558$_{\pm 1.31}$	0.0314$_{\pm 0.43}$	0.0461$_{\pm 0.96}$	<u>0.1215</u>$_{\pm 0.68}$	+23.46
5.0	**0.1210**$_{\pm 1.01}$	<u>0.1148</u>$_{\pm 0.63}$	0.0367$_{\pm 0.24}$	0.0497$_{\pm 0.91}$	0.0759$_{\pm 0.47}$	0.0083$_{\pm 0.72}$	0.0324$_{\pm 0.35}$	0.0226$_{\pm 0.56}$	0.0211$_{\pm 0.82}$	0.0804$_{\pm 0.39}$	+5.40
6.0	**0.1008**$_{\pm 0.81}$	<u>0.0853</u>$_{\pm 0.42}$	0.0317$_{\pm 0.69}$	0.0389$_{\pm 0.33}$	0.0567$_{\pm 0.94}$	0.0083$_{\pm 0.57}$	0.0226$_{\pm 0.26}$	0.0234$_{\pm 0.73}$	0.0179$_{\pm 1.18}$	0.0477$_{\pm 1.13}$	+18.17
7.0	**0.0958**$_{\pm 0.30}$	<u>0.0951</u>$_{\pm 0.47}$	0.0274$_{\pm 0.16}$	0.0298$_{\pm 0.62}$	0.0510$_{\pm 0.34}$	0.0080$_{\pm 0.99}$	0.0179$_{\pm 0.53}$	0.0190$_{\pm 0.21}$	0.0114$_{\pm 0.41}$	0.0400$_{\pm 0.67}$	+0.74
8.0	**0.0879**$_{\pm 0.32}$	<u>0.0785</u>$_{\pm 0.71}$	0.0233$_{\pm 0.45}$	0.0286$_{\pm 0.59}$	0.0428$_{\pm 0.83}$	0.0080$_{\pm 0.27}$	0.0121$_{\pm 0.56}$	0.0136$_{\pm 0.91}$	0.0076$_{\pm 0.64}$	0.0349$_{\pm 0.24}$	+11.97
9.0	**0.0780**$_{\pm 0.52}$	<u>0.0633</u>$_{\pm 0.34}$	0.0186$_{\pm 0.71}$	0.0216$_{\pm 0.76}$	0.0373$_{\pm 0.63}$	0.0073$_{\pm 0.28}$	0.0098$_{\pm 0.47}$	0.0111$_{\pm 0.98}$	0.0078$_{\pm 0.76}$	0.0278$_{\pm 0.41}$	+23.22

A.4.3 More analysis on RQ4

In the main body of the paper, Figure 2 presents the Recall@20 results on the semi-synthetic KuaiRec dataset using LightGCN++ as the backbone under varying levels of popularity bias. To facilitate comparison across all metrics at each severity level s, Table 7 reports the corresponding Recall@20, NDCG@20, and HR@20 results. Consistent with the Recall@20 findings discussed earlier, DPAA outperforms the baselines across all three metrics under both mild and severe bias.

A.4.4 Computational complexity analysis

We empirically evaluate the computational cost of DPAA by comparing it with the LightGCN backbone, since DPAA is applied on top of LightGCN. This comparison quantifies the additional training cost introduced by popularity-bias correction relative to the standard LightGCN model. Our method does not require training an additional pre-trained model, as it can reuse the backbone model from

Table 7: Performance comparison of different methods using LightGCN++ as the backbone on the semi-synthetic KuaiRec dataset under varying levels of popularity bias severity (s). We include the LightGCN++ and APDA, the best-performing baseline, for comparison. Bold and underline indicate the best and second-best results. Results are reported as mean ± std%. Improvement (%) denotes the relative gain of DPAA over the best performing method.

Metric	Method	Popularity Bias Severity (s)									
		0	1	2	3	4	5	6	7	8	9
Recall	DPAA (Ours)	$\underline{0.1953}_{\pm 0.79}$	$\mathbf{0.1999}_{\pm 0.68}$	$\mathbf{0.1814}_{\pm 0.81}$	$\mathbf{0.1354}_{\pm 0.74}$	$\mathbf{0.1011}_{\pm 0.92}$	$\mathbf{0.0950}_{\pm 0.77}$	$\mathbf{0.0919}_{\pm 1.36}$	$\mathbf{0.0810}_{\pm 1.02}$	$\mathbf{0.0649}_{\pm 0.79}$	$\mathbf{0.0584}_{\pm 1.08}$
	APDA	$0.0286_{\pm 0.61}$	$0.0155_{\pm 0.79}$	$0.0222_{\pm 0.67}$	$0.0517_{\pm 0.91}$	$0.0705_{\pm 1.33}$	$\underline{0.0799}_{\pm 1.04}$	$\underline{0.0788}_{\pm 0.82}$	$\underline{0.0747}_{\pm 1.11}$	$\underline{0.0592}_{\pm 0.76}$	$\underline{0.0519}_{\pm 1.21}$
	LightGCN++	$\mathbf{0.1954}_{\pm 0.93}$	$\underline{0.1990}_{\pm 0.71}$	$\underline{0.1803}_{\pm 0.84}$	$\underline{0.1344}_{\pm 0.67}$	$\underline{0.0929}_{\pm 1.05}$	$0.0652_{\pm 1.23}$	$0.0488_{\pm 0.89}$	$0.0401_{\pm 1.16}$	$0.0322_{\pm 0.78}$	$0.0374_{\pm 0.96}$
	Improvement (%)	-0.05	+0.45	+0.61	+0.74	+8.83	+18.90	+16.62	+8.43	+9.63	+12.52
NDCG	DPAA (Ours)	$\underline{0.1721}_{\pm 0.58}$	$\mathbf{0.2093}_{\pm 0.73}$	$\mathbf{0.1881}_{\pm 0.66}$	$\mathbf{0.1509}_{\pm 0.81}$	$\mathbf{0.1110}_{\pm 0.94}$	$\mathbf{0.1043}_{\pm 0.69}$	$\mathbf{0.0989}_{\pm 0.87}$	$\mathbf{0.0879}_{\pm 0.76}$	$\mathbf{0.0735}_{\pm 1.01}$	$\mathbf{0.0660}_{\pm 0.83}$
	APDA	$0.0359_{\pm 0.62}$	$0.0264_{\pm 0.77}$	$0.0315_{\pm 0.69}$	$0.0642_{\pm 0.88}$	$0.0811_{\pm 0.74}$	$\underline{0.0887}_{\pm 1.06}$	$\underline{0.0881}_{\pm 0.81}$	$\underline{0.0813}_{\pm 1.12}$	$\underline{0.0694}_{\pm 0.72}$	$\underline{0.0597}_{\pm 0.91}$
	LightGCN++	$\mathbf{0.1728}_{\pm 0.85}$	$\underline{0.2084}_{\pm 0.64}$	$\underline{0.1872}_{\pm 0.79}$	$\underline{0.1501}_{\pm 0.97}$	$\underline{0.1104}_{\pm 0.71}$	$0.0829_{\pm 0.82}$	$0.0635_{\pm 0.93}$	$0.0514_{\pm 1.08}$	$0.0403_{\pm 0.75}$	$0.0479_{\pm 1.03}$
	Improvement (%)	-0.41	+0.43	+0.48	+0.53	+0.54	+17.59	+12.26	+8.12	+5.91	+10.55
HR	DPAA (Ours)	$\underline{0.8835}_{\pm 1.36}$	$\mathbf{0.9314}_{\pm 1.39}$	$\mathbf{0.9060}_{\pm 1.48}$	$\mathbf{0.8223}_{\pm 1.13}$	$\mathbf{0.7047}_{\pm 0.97}$	$\mathbf{0.6968}_{\pm 1.09}$	$\mathbf{0.6681}_{\pm 1.02}$	$\mathbf{0.6163}_{\pm 1.25}$	$\mathbf{0.5525}_{\pm 1.09}$	$\mathbf{0.5119}_{\pm 1.38}$
	APDA	$0.3626_{\pm 1.28}$	$0.3193_{\pm 0.84}$	$0.3506_{\pm 1.51}$	$0.5131_{\pm 1.01}$	$0.5883_{\pm 0.79}$	$\underline{0.6224}_{\pm 1.32}$	$\underline{0.6181}_{\pm 0.87}$	$\underline{0.5926}_{\pm 1.13}$	$\underline{0.5230}_{\pm 1.42}$	$\underline{0.4946}_{\pm 1.46}$
	LightGCN++	$\mathbf{0.8843}_{\pm 1.41}$	$\underline{0.9288}_{\pm 0.75}$	$\underline{0.9013}_{\pm 0.86}$	$\underline{0.8165}_{\pm 0.72}$	$\underline{0.6913}_{\pm 1.07}$	$0.5656_{\pm 1.43}$	$0.4670_{\pm 1.18}$	$0.4159_{\pm 0.94}$	$0.3783_{\pm 1.06}$	$0.4131_{\pm 0.89}$
	Improvement (%)	-0.09	+0.28	+0.52	+0.71	+1.94	+11.95	+8.09	+4.00	+5.64	+3.50

a previous training stage, such as an already deployed or base recommendation model, as the pre-trained model. Its additional overhead arises only from the weighting mechanism during message passing. IIW estimation requires computing cosine similarity for each observed interaction, which incurs constant-time computation per interaction, while LW adds only a scalar multiplication at each layer. Therefore, DPAA does not introduce any expensive auxiliary operation and retains the same overall computational order as LightGCN, with only additional constant-factor weighting operations during message passing.

For each observed user–item interaction, computing IIW using cosine similarity between d-dimensional embeddings requires $O(d)$. Across $|\mathcal{A}|$ interactions and L layers, the total cost is $O(dL|\mathcal{A}|)$. When IIW is applied only at the initial layer, as in our default setting, this reduces to $O(d|\mathcal{A}|)$. LW requires only $O(1)$ computation per layer because it involves only scalar multiplication during aggregation. Therefore, under our default setting, the overall additional overhead per epoch is $O(d|\mathcal{A}|)$. We report the average runtime per epoch for DPAA and its LightGCN backbone in Table 8. On Coat, runtime increases from 1.09 to 1.25 seconds/epoch, while on Yahoo! R3 it increases from 1.36 to 3.16 seconds/epoch. On KuaiRec, runtime increases from 20.16 to 75.23 seconds/epoch. Overall, DPAA introduces a manageable increase in runtime for bias correction while maintaining the same overall computational order as the standard LightGCN backbone.

Table 8: Average training time in seconds per epoch for LightGCN and DPAA across datasets.

Dataset	LightGCN (seconds/epoch)	DPAA (seconds/epoch)
Coat	1.09	1.25
KuaiRec	20.16	75.23
Yahoo! R3	1.36	3.16

A.5 Ablation Studies

A.5.1 Contribution of different components of DPAA

We analyze the contribution of different components in DPAA with the KuaiRec semi-synthetic clicks under varying levels of popularity bias severity (s). We show the results in Figure 6. By default, DPAA uses $\gamma = 1$, meaning that IIW is applied only at the first layer.

First, comparing DPAA, which applies IIW only in the initial layer, with DPAA-γ=0, where IIW is applied to all layers, we observe that DPAA performs better in most cases. This suggests that restricting IIW to the initial layer is generally effective, since applying it across all layers may impose overly aggressive correction and remove useful higher-order collaborative signals.

Next, we examine the role of combining pre-trained (m) and current (c) embeddings for IIW estimation. To do so, we consider two variants: one that excludes the pre-trained model estimate

and another that excludes the current model estimate when computing IIW. Removing either the pre-trained estimate (DPAA-w/o-m and DPAA-γ=0-w/o-m) or the current estimate (DPAA-w/o-c and DPAA-γ=0-w/o-c) generally degrades performance compared with DPAA. This indicates that relying solely on either pre-trained or current embeddings is insufficient, whereas combining them through a smooth transition from the pre-trained estimate to the dynamic model estimate provides a more stable and accurate IIW estimation.

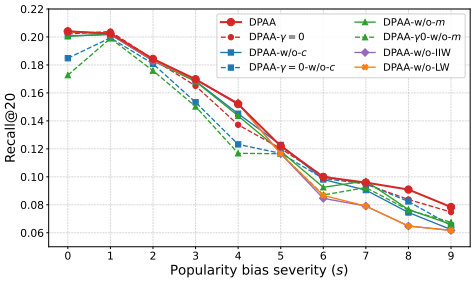


Figure 6: Performance comparison of DPAA variants for Recall@20 on the KuaiRec dataset by varying levels of popularity bias severity (s), showing the contribution of different components.

Finally, removing the other key components, IIW and LW, leads to clear performance drops. DPAA-w/o-IIW typically underperforms DPAA, highlighting the importance of interaction-level weighting in reducing popularity bias. Similarly, DPAA-w/o-LW shows degraded performance compared to DPAA, indicating that layer-wise weighting is crucial for mitigating shallow-layer dominance and preserving long-range collaborative signals.

Overall, these results demonstrate that both IIW and LW are essential components of DPAA. They also show that combining pre-trained and current embeddings through a smooth transition is important for reliable IIW estimation and improved performance. Together, these findings highlight the importance of all key components of our method in reducing popularity bias and improving recommendation performance. We also provide the results in Table 9.

Table 9: Performance comparison of DPAA variants for Recall@20 on the KuaiRec dataset by varying levels of popularity bias severity (s), showing the contribution of different components. Bold and underline indicate the best and second-best results.

s	DPAA	DPAA-γ=0	DPAA-w/o-c	DPAA-γ=0-w/o-c	DPAA-w/o-m	DPAA-γ=0-w/o-m	DPAA-w/o-IIW	DPAA-w/o-LW
0.0	<u>0.2040</u>	0.2021	0.2007	0.1848	0.2005	0.1727	**0.2043**	0.2036
1.0	<u>0.2025</u>	**0.2039**	0.2017	0.1994	0.2021	0.1990	0.2019	0.2024
2.0	**0.1843**	<u>0.1841</u>	0.1833	0.1808	0.1829	0.1759	0.1829	0.1836
3.0	**0.1698**	0.1650	0.1685	0.1533	0.1682	0.1503	<u>0.1695</u>	0.1693
4.0	0.1520	0.1372	0.1451	0.1232	0.1436	0.1168	**0.1528**	<u>0.1524</u>
5.0	<u>0.1220</u>	0.1202	**0.1228**	0.1168	0.1183	0.1165	0.1165	0.1165
6.0	<u>0.0999</u>	**0.1005**	0.0981	0.0988	0.0925	0.0870	0.0846	0.0867
7.0	<u>0.0959</u>	0.0947	0.0905	0.0950	**0.0970**	0.0923	0.0791	0.0791
8.0	**0.0909**	<u>0.0838</u>	0.0745	0.0824	0.0766	0.0763	0.0648	0.0648
9.0	**0.0784**	<u>0.0748</u>	0.0623	0.0652	0.0660	0.0675	0.0617	0.0617

A.5.2 Sensitivity analysis of the pre-trained model

We further examine the sensitivity of DPAA to the quality of the pre-trained model used to estimate IIW. We consider LightGCN models trained for 20 epochs, 50 epochs, and until full convergence, where fewer epochs generally correspond to less well-trained embeddings and thus lower-quality pre-trained embeddings. As shown in Table 10, DPAA remains reasonably robust even when the quality of the pre-trained model is lower. On Coat, Recall@20 and NDCG@20 remain unchanged at 0.2565 and 0.2065, respectively, across 20 epochs, 50 epochs, and full convergence. On KuaiRec, Recall@20 improves from 0.0280 at 20 epochs to 0.0301 at 50 epochs and 0.0349 with full convergence; NDCG@20 follows a similar improvement from 0.0507 to 0.0530 and 0.0586. On Yahoo! R3, the differences are smaller, with Recall@20 increasing from 0.1442 to 0.1453 and 0.1479, and NDCG@20 increasing from 0.0653 to 0.0655 and 0.0659. Overall, better-quality pre-trained models generally improve DPAA's performance, while lower-quality pre-trained embeddings do not cause substantial degradation. Therefore, these results demonstrate that DPAA is not highly sensitive to the quality of the pre-trained model. This robustness is partly because IIW does not rely solely on the pre-trained model, but also incorporates the interaction signal learned by the current model during training.

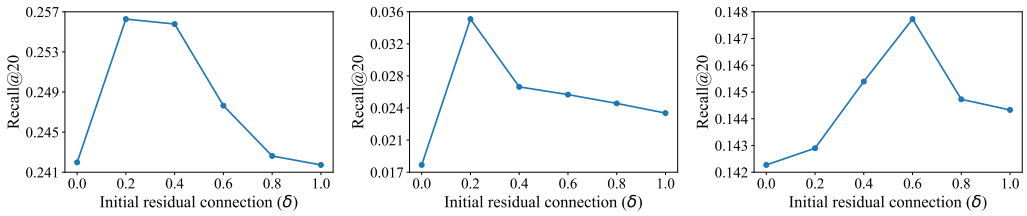
Table 10: Sensitivity analysis of DPAA to the quality of the pre-trained model across different datasets. We use pre-trained LightGCN models after 20 epochs, 50 epochs, and full convergence to estimate the IIW in our DPAA method. Bold and underline indicate the best and second-best results.

Pre-trained model	Coat		KuaiRec		Yahoo! R3	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
20 epochs	**0.2565**	**0.2065**	0.0280	0.0507	0.1442	0.0653
50 epochs	**0.2565**	**0.2065**	<u>0.0301</u>	<u>0.0530</u>	<u>0.1453</u>	<u>0.0655</u>
Fully converged	**0.2565**	**0.2065**	**0.0349**	**0.0586**	**0.1479**	**0.0659**

A.5.3 Parameter sensitivity analysis

To examine the sensitivity of DPAA to its key hyperparameters, we conduct additional experiments by varying the initial residual connection parameter δ, the layer-wise weighting parameter η, and the IIW transition parameter C. We evaluate performance on the Coat, KuaiRec, and Yahoo! R3 datasets. The parameter search ranges follow the implementation settings described in Section A.3.7.

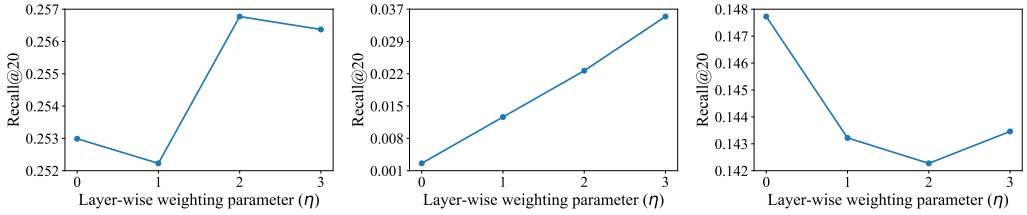
Effect of the initial residual connection. Figure 7 shows the effect of varying δ, which controls the contribution of the initial node embeddings relative to neighborhood aggregation. The results indicate that retaining a moderate amount of initial embedding information is generally beneficial. On Coat, Recall@20 increases substantially from $\delta = 0$ and reaches its highest value around $\delta = 0.2$, after which performance gradually decreases. KuaiRec exhibits a similar pattern, with the best performance at $\delta = 0.2$. On Yahoo! R3, performance improves up to a larger value of δ, reaching its maximum around $\delta = 0.6$. These results suggest that the optimal amount of residual information depends on the characteristics of each dataset. In particular, preserving initial embeddings can help prevent over-smoothing while still allowing message passing to incorporate higher-order collaborative signals.



(a) Coat dataset (b) KuaiRec dataset (c) Yahoo! R3 dataset

Figure 7: Effect of varying the *initial residual connection* ($\boldsymbol{\delta}$) across different datasets.

Effect of layer-wise weighting. Figure 8 evaluates the layer-wise weighting parameter η, which determines how strongly higher-order representations are emphasized. The effect varies across datasets. On Coat, performance is highest at $\eta = 2$, indicating that emphasizing deeper layers can improve recommendation quality even under relatively mild popularity bias. KuaiRec shows a stronger benefit from increasing η, with Recall@20 increasing consistently across the evaluated range and reaching its highest value at $\eta = 3$. This is consistent with the stronger popularity skew

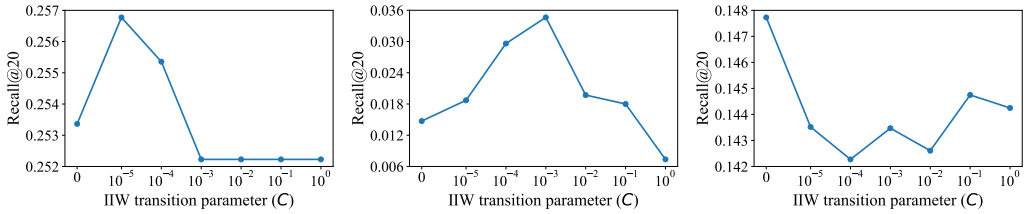


(a) Coat dataset (b) KuaiRec dataset (c) Yahoo! R3 dataset

Figure 8: Effect of varying the layer-wise weighting parameter ($\boldsymbol{\eta}$) across different datasets.

in KuaiRec, where higher-order collaborative signals can help reduce the dominance of shallow, popularity-driven interactions. In contrast, Yahoo! R3 achieves its best performance at $\eta = 0$, suggesting that uniformly weighting the layers is preferable in this sparse setting.

Effect of the IIW transition parameter. Figure 9 studies C, which controls the transition between the pre-trained and current-model embeddings used to estimate IIW. Performance varies for different values of C on different datasets. On Coat, a small but nonzero value of C provides the strongest performance, with $C = 10^{-5}$ selected as the best configuration. On KuaiRec, performance improves as C increases from very small values and reaches its maximum around $C = 10^{-3}$, after which it decreases. This suggests that combining stable pre-trained representations with adaptive current-model representations can provide a more reliable interaction-strength estimate than relying exclusively on either source. On Yahoo! R3, $C = 0$ performs best, indicating that the pre-trained embeddings are more useful for IIW estimation in this sparse dataset, where current embeddings may be less reliable.



(a) Coat dataset (b) KuaiRec dataset (c) Yahoo! R3 dataset

Figure 9: Effect of varying the IIW transition parameter (C) across different datasets.

Overall, the results demonstrate that the optimal values of δ, η, and C vary across datasets. This variation reflects differences in popularity skew and interaction sparsity. The results further support the design choices of DPAA: retaining initial embeddings helps mitigate over-smoothing, layer-wise weighting controls the contribution of higher-order collaborative signals, and the transition mechanism balances stable pre-trained representations with adaptive current-model representations.