PDFlow: Popularity-Debiased Flow Matching for Sequential Recommendation

Zican Yang[1] , **Zeyu Li**[1] , **Cong He**[1] , **Xiaotao Wu**[1] , **Guanfeng Liu**[2] and **Pengpeng Zhao**[1*]
[1]School of Computer Science and Technology, Soochow University, China
[2]School of Computing, Macquarie University, Australia
{20255227036, zylisuda, chesuda, 20245227017}@stu.suda.edu.cn, guanfeng.liu@mq.edu.au, ppzhao@suda.edu.cn

Abstract

Generative models have emerged as a powerful paradigm in sequential recommendation due to their superior distribution modeling. However, long-tail data distributions inevitably induce popularity bias, as iterative generation trajectories gravitate toward dense clusters of popular items. Current debiasing methods struggle to enforce consistent step-wise constraints, rendering them ineffective for the multi-step process of generative recommendation. To address this challenge, we propose a method called **P**opularity-**D**ebiased **Flow** Matching for sequential recommendation (PDFlow). Specifically, grounded in the flow matching framework, PDFlow integrates *a residual popularity vector field* beyond the main flow to model the popularity debiasing flow at each step. This enables real-time trajectory intervention to adjust the generative path, preventing the model from overemphasizing popular items. Subsequently, we incorporate *cross-popularity alignment loss*, using co-occurrence patterns within the same sequence to align features of both popular and tail items, thereby mitigating popularity-driven distribution separation. Jointly, trajectory correction prevents popularity overfitting while cross-popularity alignment ensures latent fairness, effectively mitigating popularity bias. Extensive experiments on four real-world datasets demonstrate the effectiveness of PDFlow in improving long-tail coverage and overall recommendation performance. All codes and datasets are available at https://github.com/eqmll/PDFlow.

1 Introduction

Sequential recommendation (SR) captures evolving users interests through sequential dependency modeling [Hidasi *et al.*, 2015; Rendle *et al.*, 2010]. Traditional methods, including RNN-based [Hidasi *et al.*, 2016], CNN-based [Tang and Wang, 2018], and self-attention models [Kang and McAuley, 2018; Sun *et al.*, 2019], rely on static point estimates, struggling to express interest uncertainty and diversity [Xie *et al.*, 2021; Wang *et al.*, 2023; Li *et al.*, 2024a; Mao *et al.*, 2025]. Consequently, recent research has shifted towards generative paradigms to capture probabilistic user behaviors. For instance, DiffuRec [Li *et al.*, 2024b] and DreamRec [Yang *et al.*, 2023b] employ Diffusion Models (DM) to model item distributions through iterative Gaussian denoising, while FMRec [Liu *et al.*, 2025] and GMFlowRec [Ye *et al.*, 2025] adopt Flow Matching (FM) to construct deterministic Ordinary Differential Equation (ODE) trajectories for efficient sampling.

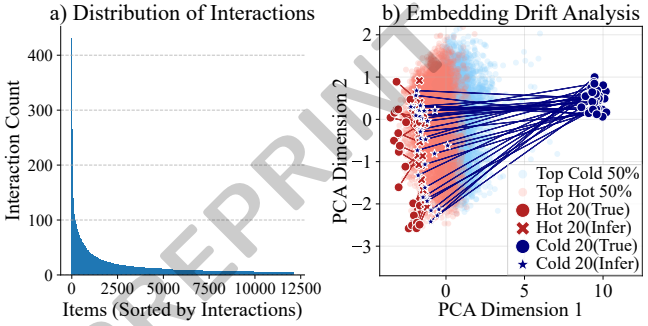


Figure 1: **Visual analysis of popularity bias on Amazon [Ni *et al.*, 2019].** (a) Item popularity exhibits a long-tail distribution. (b) PCA visualisation of FMRec [Liu *et al.*, 2025] embeddings clustered into two groups. Dots denote actual positions, with red crosses and blue asterisks indicating predicted locations for top-20 popular (red) and long-tail (blue) items, respectively.

However, these generative approaches remain susceptible to data imbalances due to their reliance on observed interaction frequencies. In sequential recommendation settings, user interactions typically follow a pronounced long-tail distribution, where a minority of popular items account for the majority of interactions. Such imbalance forces the model to over-memorize popular patterns during training [Qin *et al.*, 2023]. During generative inference, this bias is further exacerbated by exposure bias [Ning *et al.*, 2023], where small deviations toward popular regions progressively accumulate and amplify at each step. Consequently, the generated trajectories inevitably gravitate toward popular items, drifting away from true personalized interests.

To uncover this bias mechanism, we analyzed FMRec [Liu *et al.*, 2025] using the Amazon-Beauty dataset. Figure 1 reveals two pathological phenomena: first, beyond the long-

* Corresponding author.

tail distribution in Figure 1(a), we observe severe popularity-driven stratification in Figure 1(b), where distinct "hot" and "cold" clusters indicate that popularity rather than semantics dominates the representation topology. Furthermore, the model exhibits systematic trajectory drift: generation paths for tail items are progressively drawn toward high-density popular regions rather than intended targets. This suggests the generative process exacerbates data imbalances via a "rich-get-richer" loop, rendering tail items inaccessible during inference.

While conventional debiasing paradigms, such as Inversion Propensity Scoring (IPS) [Schnabel *et al.*, 2016] and causal inference [Ning *et al.*, 2024; Tan *et al.*, 2025], offer established solutions for discriminative models, their adaptation to generative recommendation remains non-trivial. Specifically, the methodological disparity between these two paradigms creates a substantial obstacle to effective debiasing, leading to two major challenges: **Challenge 1: Static weights are insufficient for governing a dynamic process.** Traditional methods mitigate bias by assigning fixed scalar weights to the training objective [Schnabel *et al.*, 2016; Cui *et al.*, 2019]. In contrast, generative sequential recommendation involves a continuous, multi-step evolutionary process. First, a static weight applied to the final result lacks the temporal granularity to intervene in intermediate steps. Second, even when constraints are imposed at each generation step, it is arduous to maintain consistent constraints and correct guidance strength across varying time steps. Consequently, existing re-weighting strategies face significant limitations in rectifying the cumulative trajectory drift. **Challenge 2: Score adjustment is inadequate for rectifying representation separation.** Conventional logit re-weighting acts merely as a post-hoc patch, failing to address the underlying representation [Kang *et al.*, 2019; Zhao *et al.*, 2023]. Fundamentally, generative inference functions as a rounding process (mapping continuous vectors to discrete top-k items) [Li *et al.*, 2024b]. However, our analysis reveals the popularity-driven stratification, where there is a distinct gap between the representations of popular and tail items. The rounding mechanism inherently pulls generated vectors towards dense popular embeddings. Consequently, scalar adjustments cannot rectify this rounding error, necessitating fundamental integration of the underlying structure.

To address these challenges, we propose Popularity-Debiased Flow Matching for Sequential Recommendation (PDFlow). Flow matching provides deterministic trajectories and explicitly models velocity fields. Building upon this framework, PDFlow incorporates two complementary mechanisms to counter popularity bias: 1) **Residual Popularity Vector Field.** We employ a "decomposition and subtraction" strategy to construct a popularity-driven residual vector field. During the generation phase, this mechanism operates as a real-time filter, actively subtracting specific gravitational components to prevent trajectories from drifting towards popular items. By counteracting the Matthew effect [Abdollahpouri *et al.*, 2019], the generated results more faithfully reflect genuine user interests. 2) **Cross-Popularity Consistency Loss.** This component leverages behavior sequences to narrow potential distribution gaps caused by frequency disparities by enforcing proximity between co-occurring popular and long-tail items. These designs enable PDFlow to rectify biased vector fields and align latent distributions, decoupling user intent from popularity bias without compromising recommendation fidelity. Finally, the main contributions of this paper are as follows:

- We propose PDFlow, a new approach to alleviate popularity bias in generative sequential recommendation.
- We integrate a Residual Popularity Vector Field with a Cross-Popularity Alignment Loss to jointly mitigate popularity bias.
- Experiments on four real-world datasets show that PDFlow outperforms existing methods in both recommendation accuracy and popularity debiasing.

2 Related Work

2.1 Sequential Recommendation

Sequential recommendation characterizes user preferences by leveraging diverse architectural paradigms, including RNN-based methods (e.g., GRU4Rec [Hidasi *et al.*, 2016]), CNN-based methods (e.g., Caser [Tang and Wang, 2018], NextItNet [Yuan *et al.*, 2019]), and Transformer-based methods (e.g., SASRec [Kang and McAuley, 2018], BERT4Rec [Sun *et al.*, 2019], TiSASRec [Li *et al.*, 2020]). These paradigms prioritize deterministic ranking representations over the inherent stochasticity of user interests.

Generative sequential recommendation models the next-item distributions through continuous representation synthesis and subsequent rounding-based retrieval. Diffusion-based methods (e.g., DiffuRec [Li *et al.*, 2024b], DreamRec [Yang *et al.*, 2023b]) iteratively denoise a noisy representation. More recently, flow-matching-based methods (e.g., FMRec [Liu *et al.*, 2025], GMFlowRec [Ye *et al.*, 2025]) learn deterministic ODE trajectories to speed up inference. Unlike previous generative recommenders, PDFlow uses residual popularity modeling with step-wise subtraction and cross-popularity alignment for bias neutralization during training and inference.

2.2 Popularity Debiasing

Popularity bias causes the over-recommendation of head items at the expense of niche content [Tan *et al.*, 2025; Liu *et al.*, 2026]. To mitigate this, reweighting strategies use IPS to correct exposure bias [Schnabel *et al.*, 2016; Joachims *et al.*, 2017], and regularization methods prevent models from overemphasizing head items by introducing penalty terms [Abdollahpouri *et al.*, 2017; Cui *et al.*, 2019].

Recently, causal inference has garnered significant attention. Through methods such as backdoor adjustment and counterfactual inference, it identifies and eliminates popularity as a confounding factor [Ning *et al.*, 2024; Tan *et al.*, 2025; Zheng *et al.*, 2021; Wei *et al.*, 2021; Zhang *et al.*, 2021]. Additionally, representation optimization aims to rectify embedding distortions through contrastive learning and decoupling mechanisms[Yang *et al.*, 2023a; Kang *et al.*, 2019]. However, these conventional paradigms are tailored for discriminative ranking and are incompatible with the continuous dynamics of generative models. To address this, we propose PDFlow, which mitigates popularity bias by rectifying the generative vector field through residual subtraction and neutralizing representation gaps via cross-popularity alignment.

3 Preliminaries

3.1 Problem Statement

For the generative sequential recommendation task, $\mathcal{U}$ and $\mathcal{I}$ are defined as the set of users and items, respectively. For each user $u \in \mathcal{U}$, their interaction history is represented as a time series $\mathcal{S}_u = [i_1, i_2, \ldots, i_L]$, where $i_\ell \in \mathcal{I}$ denotes the item of the ℓ-th interaction. The objective is to predict the most probable next item given $\mathcal{S}_u$. To map discrete items to dense vectors, we employ an embedding table $\mathbf{E} \in \mathbb{R}^{|\mathcal{I}| \times d}$. Here, $\mathbf{e}_i \in \mathbb{R}^d$ denotes the embedding vector for item i. This paradigm first generates a continuous representation $\mathbf{x}^* \in \mathbb{R}^d$ through an iterative generation process, followed by rounding-based search using inner product similarity to obtain the final recommendation result:

$$\text{score}(j) = {\mathbf{x}^*}^\top \mathbf{e}_j, \qquad \hat{\mathcal{R}}_k = \text{topk}_{j \in \mathcal{I}} \, \text{score}(j). \quad (1)$$

Here, $\hat{\mathcal{R}}_k$ denotes the top k item recommendation set based on similarity scores. However, the long-tail distribution segregates $\mathbf{E}$ according to popularity and draws $\mathbf{x}^*$ towards the popular region. Furthermore, rounding exacerbates this distortion by marginalizing tail items.

3.2 Flow Matching

Flow Matching [Lipman *et al.*, 2023] aims to construct a probability path that transports a simple prior distribution to the complex data distribution.

Forward Process. Let $\mathbf{x}_1 \sim p_{data}$ and $\mathbf{x}_0 \sim p_{prior}$ denote a clean data sample and a noise sample, respectively. We define a time-dependent probability path via a linear interpolation mapping $\boldsymbol{\psi}_t$:

$$\mathbf{x}_t = \boldsymbol{\psi}_t(\mathbf{x}_1, \mathbf{x}_0) = (1-t)\mathbf{x}_1 + t\mathbf{x}_0, \quad t \in [0,1], \quad (2)$$

where $t = 0$ corresponds to $\mathbf{x}_1$ and $t = 1$ to $\mathbf{x}_0$. This displacement map defines an Optimal Transport (OT) path [Liu *et al.*, 2023]. The associated conditional vector field $\mathbf{u}_t(\mathbf{x}_t|\mathbf{x}_1, \mathbf{x}_0)$, which generates this trajectory, is the time derivative of $\boldsymbol{\psi}_t$:

$$\mathbf{u}_t(\mathbf{x}_t|\mathbf{x}_1, \mathbf{x}_0) = \frac{d}{dt}\boldsymbol{\psi}_t(\mathbf{x}_1, \mathbf{x}_0) = \mathbf{x}_0 - \mathbf{x}_1, \quad (3)$$

A neural network $\mathbf{v}_\Theta(t, \mathbf{x}_t)$ parameterized by Θ is trained to regress this field. Following the Conditional Flow Matching (CFM) framework, the objective $\mathcal{L}_{FM}(\Theta)$ is formulated as:

$$\begin{aligned}\mathcal{L}_{FM}(\Theta) &= \mathbb{E}_{t,\mathbf{x}_1,\mathbf{x}_0}\|\mathbf{v}_\Theta(\mathbf{x}_t, t) - \mathbf{u}_t(\mathbf{x}_t|\mathbf{x}_1, \mathbf{x}_0)\|^2 \\ &= \mathbb{E}_{t,\mathbf{x}_1,\mathbf{x}_0}\|\mathbf{v}_\Theta(\mathbf{x}_t, t) - (\mathbf{x}_0 - \mathbf{x}_1)\|^2, \end{aligned} \quad (4)$$

where $t \sim \mathcal{U}[0,1]$, $\mathbf{x}_1 \sim p_{\text{data}}$, $\mathbf{x}_0 \sim p_{\text{prior}}$, and $\|\cdot\|_2$ denotes the Euclidean norm.

Reverse Process. During inference, data generation is formulated as an ODE. Since $\mathbf{v}_\Theta$ describes the forward direction ($t: 0 \to 1$), the inverse generation process follows:

$$d\mathbf{x} = -\mathbf{v}_\Theta(\mathbf{x}_t, t)dt. \quad (5)$$

The sample $\mathbf{x}_1$ is synthesized by integrating the vector field $\mathbf{v}_\Theta$ backward in time ($t: 1 \to 0$) starting from $\mathbf{x}_0$.

4 Methodology

This section details the forward training process and reverse inference process of PDFlow. During forward process, PDFlow models the overall generation dynamics through two components: a Main Flow Vector Field for global user interest, and a Residual Popularity Vector Field that counteracts trajectory drift by regressing towards a popularity center. During backward inference, PDFlow employs Eulerian sampling based on ODE, subtracting predicted popularity components from the vector field at each step.

4.1 Main Flow Field Modeling

PDFlow builds upon standard FM sequence recommendation models. We first define the main flow vector field before establishing a foundation for our debiasing mechanisms.

Noise Embedding Construction. For a user sequence $\mathcal{S}_u = [i_1, \ldots, i_L]$ corresponds to the item embedding $\mathbf{e}_\ell = \text{Emb}(i_\ell) \in \mathbb{R}^d$, forming the sequence representation matrix $\mathbf{E}_u = [\mathbf{e}_1, \ldots, \mathbf{e}_L] \in \mathbb{R}^{L \times d}$. Set i_{L+1} as the next target item, whose embedding is defined as $\mathbf{e}_{L+1} = \text{Emb}(i_{L+1})$. This vector is sampled from the true item distribution p_c. Then, a straight flow trajectory is constructed by interpolation between the target embedding $\mathbf{e}_{L+1}$ and Gaussian noise $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$\mathbf{x}_t = (1-t)\,\mathbf{e}_{L+1} + t\,\mathbf{z}_0, \qquad t \in [0,1]. \quad (6)$$

where $\mathbf{I} \in \mathbb{R}^{d \times d}$ denotes the identity matrix. We adopt the mode sampling technique with heavy tails [Esser *et al.*, 2024] to sample the time steps t.

Subsequently, we integrate the noisy state $\mathbf{x}_t$ and timestep t into each vector within the sequence representation matrix $\mathbf{E}_u$ as the input for the vector field estimator. Following the integration mechanism proposed in [Li *et al.*, 2024b], the fusion process for each item embedding $\mathbf{e}_\ell$ is defined as:

$$\mathbf{h}_\ell = \mathbf{e}_\ell + \boldsymbol{\lambda}_\ell \odot \big(\mathbf{x}_t + \text{emb}(t)\big), \quad (7)$$

where $\odot$ denotes the Hadamard product, and $\text{emb}(t) \in \mathbb{R}^d$ is the time embedding. $\boldsymbol{\lambda}_\ell$ is a gating vector sampled from $\mathcal{N}(\delta, \delta)$. We set $\mathbf{H}_u = [\mathbf{h}_1, \ldots, \mathbf{h}_L]$ as the noise embeddings.

Main Flow Vector Field. Based on the noise embedding $\mathbf{H}_u$, we can construct the vector field estimator. Following [Liu *et al.*, 2025], we opt to predict the target item embedding $\hat{\mathbf{x}}$ instead of directly regressing the vector field, as this strategy proves more effective for recommendation. The vector field $\hat{\mathbf{v}}$ can be easily derived from $\hat{\mathbf{x}}$ via Equation 3 as:

$$\hat{\mathbf{v}} = \mathbf{z}_0 - \hat{\mathbf{x}}. \quad (8)$$

Let $\text{LAST}([\mathbf{q}_1, \ldots, \mathbf{q}_L]) = \mathbf{q}_L$ denote the operator that takes the last-token representation. Using $\mathbf{H}_u$, $\mathbf{x}_t$, and t as inputs, the Transformer employs the final token's representation as the predicted target item vector:

$$\hat{\mathbf{x}} = f_\theta(\mathbf{x}_t, t, \mathbf{E}_u) = \text{LAST}\big(\text{Trm}_1(\mathbf{H}_u)\big). \quad (9)$$

By substituting Eq. (8) and Eq. (9) into Eq. (4), the flow matching objective can be reformulated as the mean squared error between the predicted and target embeddings:

$$\begin{aligned}\mathcal{L}_{\text{FM}}^{\text{main}} &= \mathbb{E}_{t,\,\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{e}_{L+1} \sim p_c}\Big[\|\hat{\mathbf{x}} - \mathbf{e}_{L+1}\|_2^2\Big] \\ &= \mathbb{E}_{t,\,\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{e}_{L+1} \sim p_c}\Big[\|f_\theta(\mathbf{x}_t, t, \mathbf{E}_u) - \mathbf{e}_{L+1}\|_2^2\Big]. \end{aligned} \quad (10)$$

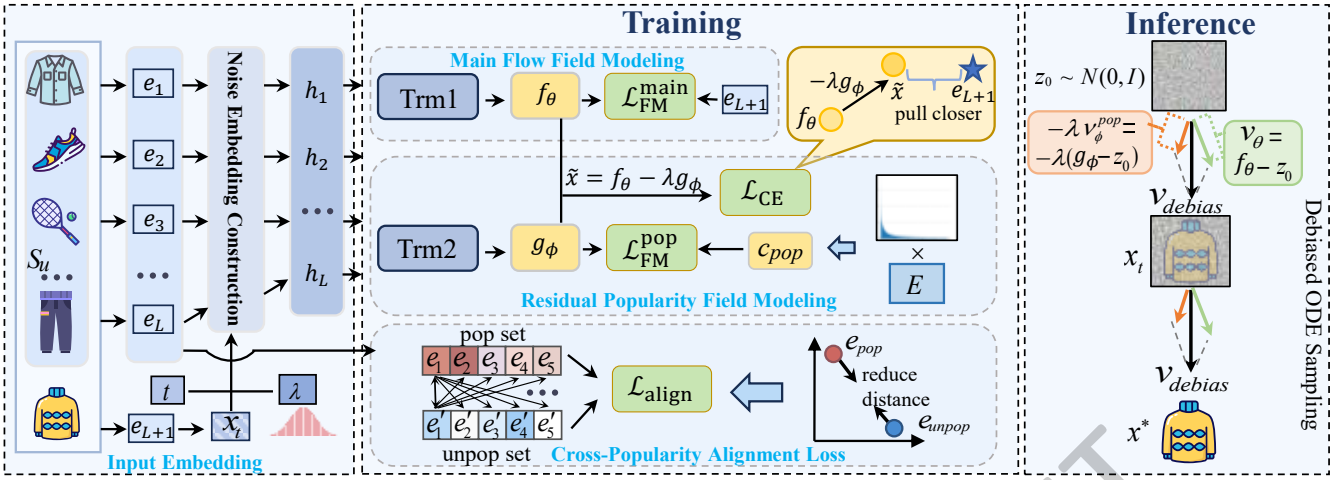


Figure 2: **PDFlow framework overview.** During training, noise-augmented sequence embeddings are processed by Transformer-based encoders Trm1 and Trm2 to predict mainstream and popularity embeddings, optimized via $\mathcal{L}_{\text{FM}}$, $\mathcal{L}_{\text{CE}}$ and the cross-popularity alignment loss $\mathcal{L}_{\text{align}}$. During inference, the debiased ODE prevents trajectory drift by subtracting the popularity component at each step.

4.2 Residual Popularity Field Modeling

To correct popularity-driven drift, we define a global popularity center as the popularity-weighted centroid of item embeddings. This center acts as a distorted terminal destination representing worst-case bias. By subtracting this target-oriented vector field bias, we can recover genuine user intent.

Global Popularity Center. The interaction count for item i in the training log is denoted as $\text{cnt}(i)$. The normalized popularity for item i is

$$p(i) = \frac{\text{cnt}(i)}{\sum_{j\in\mathcal{I}} \text{cnt}(j)}. \tag{11}$$

Based on $p(i)$, the global popularity center is defined as:

$$\mathbf{c}_{\text{pop}} = \sum_{i\in\mathcal{I}} p(i)\mathbf{e}_i. \tag{12}$$

Residual Popularity Vector Field. To synchronously estimate the vector field of the trajectory drifting toward the worst-case destination $\mathbf{c}_{\text{pop}}$, we introduce a popularity Transformer to predict the popularity-driven item embedding:

$$\hat{\mathbf{x}}_{\text{pop}} = g_\phi(\mathbf{x}_t, t, \mathbf{E}_u) = \text{LAST}\big(\text{Trm}_2(\mathbf{H}_u)\big). \tag{13}$$

Following the form of Eq.10, the FM loss for this component regresses the predicted popularity-driven embedding toward the global popularity center:

$$\begin{aligned}\mathcal{L}_{\text{FM}}^{\text{pop}} &= \mathbb{E}_{t,\,\mathbf{z}_0\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{c}_{\text{pop}}}\left[\|\hat{\mathbf{x}}_{\text{pop}} - \mathbf{c}_{\text{pop}}\|_2^2\right] \\ &= \mathbb{E}_{t,\,\mathbf{z}_0\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{c}_{\text{pop}}}\left[\|g_\phi(\mathbf{x}_t, t, \mathbf{E}_u) - \mathbf{c}_{\text{pop}}\|_2^2\right].\end{aligned} \tag{14}$$

For simplicity, the combined FM loss can be described as

$$\mathcal{L}_{\text{FM}} = \mathcal{L}_{\text{FM}}^{\text{main}} + \mathcal{L}_{\text{FM}}^{\text{pop}}. \tag{15}$$

Cross-Entropy Loss. Using the obtained $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}_{\text{pop}}$, we compute a cross-entropy loss to provide additional supervisory signals for precise item retrieval. This loss pulls the debiased item embedding $\tilde{\mathbf{x}}$ toward the target $\mathbf{e}_{L+1}$ while pushing it away from other item embeddings $\{\mathbf{e}_j\}_{j\neq i_{L+1}}$. The cross-entropy loss is computed as:

$$\mathcal{L}_{\text{CE}} = -\log \frac{\exp(\tilde{\mathbf{x}}^\top \mathbf{e}_{L+1})}{\sum_{j\in\mathcal{I}} \exp(\tilde{\mathbf{x}}^\top \mathbf{e}_j)}. \tag{16}$$

The debiased embedding $\tilde{\mathbf{x}}$ is computed by subtracting the residual popularity component from $\hat{\mathbf{x}}$:

$$\tilde{\mathbf{x}} = \hat{\mathbf{x}} - \lambda\,\hat{\mathbf{x}}_{\text{pop}}, \tag{17}$$

where $\lambda \in [0, 1]$ is a scalar hyperparameter controlling the subtraction strength.

4.3 Overall Training Objective

Cross-Popularity Alignment Loss. Leveraging the principle that items within the same user sequence share similar latent semantics [Yang *et al.*, 2023a], we introduce cross-popularity alignment loss to bridge popularity induced representation gaps. Specifically, we partition $\mathcal{S}_u$ into popular and unpopular subsets using the global median popularity threshold: $\tau = \text{median}(\{p(i) : i \in \mathcal{I}\})$. Items are then split into popular and unpopular sets as

$$\begin{aligned}\mathcal{P}_u^{\text{pop}} &= \{i \in \tilde{\mathcal{S}}_u \mid p(i) > \tau\}, \\ \mathcal{P}_u^{\text{unpop}} &= \{i \in \tilde{\mathcal{S}}_u \mid p(i) \le \tau\}.\end{aligned} \tag{18}$$

The cross-popularity alignment loss is computed as:

$$\mathcal{L}_{\text{align}} = \mathbb{E}_u\left[\frac{1}{|\mathcal{P}_u^{\text{pop}}|\,|\mathcal{P}_u^{\text{unpop}}|}\sum_{i\in\mathcal{P}_u^{\text{pop}}}\sum_{j\in\mathcal{P}_u^{\text{unpop}}}\|\mathbf{e}_i - \mathbf{e}_j\|_2^2\right]. \tag{19}$$

Overall Objective. The final training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{FM}} + \alpha\,\mathcal{L}_{\mathrm{CE}} + \beta\,\mathcal{L}_{\mathrm{align}}, \tag{20}$$

where α and β balance retrieval consistency and cross-popularity alignment.

4.4 Debiased ODE Sampling at Inference

During inference, sampling follows the ODE in Subsection. 3.2 and is solved by the Euler method. Given $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, initialize $\mathbf{x}_0 = \mathbf{z}_0$. Time is discretized into N steps with

$$t \in \left\{0, \tfrac{1}{N}, \ldots, \tfrac{N-1}{N}, 1\right\}, \qquad \Delta t = \tfrac{1}{N}. \tag{21}$$

The debiased generative dynamics are defined by the ODE

$$d\mathbf{x}_t = \mathbf{v}_{\mathrm{debias}}(\mathbf{x}_t, t, \mathbf{E}_u)\,dt, \qquad t \in [0, 1]. \tag{22}$$

At each discrete time step t, the embedding predictors output $\hat{\mathbf{x}} = f_\theta(\mathbf{x}_t, t, \mathbf{H}_u)$ and $\hat{\mathbf{x}}_{\mathrm{pop}} = g_\phi(\mathbf{x}_t, t, \mathbf{H}_u)$, which are used to compute the debiased vector field $\mathbf{v}_{\mathrm{debias}}(\mathbf{x}_t, t)$. The vector fields are negated to reflect that inference follows the reverse trajectory of training, defined as:

$$\begin{aligned} \mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{E}_u) &= f_\theta(\mathbf{x}_t, t, \mathbf{E}_u) - \mathbf{z}_0, \\ \mathbf{v}_\phi^{\mathrm{pop}}(\mathbf{x}_t, t, \mathbf{E}_u) &= g_\phi(\mathbf{x}_t, t, \mathbf{E}_u) - \mathbf{z}_0, \end{aligned} \tag{23}$$

and the debiased vector field is

$$\mathbf{v}_{\mathrm{debias}}(\mathbf{x}_t, t, \mathbf{E}_u) = \mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{E}_u) - \lambda\,\mathbf{v}_\phi^{\mathrm{pop}}(\mathbf{x}_t, t, \mathbf{E}_u). \tag{24}$$

The λ here is the same hyperparameter as in Eq. (17), ensuring consistent debiasing strength during both training and inference. The ODE in Eq. (22) is solved by Euler updates

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t \cdot \mathbf{v}_{\mathrm{debias}}(\mathbf{x}_t, t, \mathbf{E}_u). \tag{25}$$

After N steps, the generated representation $\mathbf{x}^*$ is obtained and rounding retrieval is performed by Eq. (1).

4.5 Complexity Analysis

We analyze the time and space complexity of PDFlow below.

Time Complexity. The computational cost of PDFlow comprises: 1) Transformer Backbone: $O(L^2 d + L d^2)$ from self-attention and feed-forward networks, 2) Cross-Popularity Alignment: $O(L^2 d)$ for pairwise distances within the sequence, and 3) Output Projection: $O(|\mathcal{I}|d)$ for full item retrieval. Consequently, the overall time complexity is: $O(N \cdot (L^2 d + L d^2) + |\mathcal{I}|d)$. Training involves a single forward pass ($N = 1$), whereas inference executes the backbone for N steps using an ODE solver.

Space Complexity. For space complexity, the item embedding table requires $O(|\mathcal{I}|d)$ space, while the Transformer weights scale with $O(d^2)$. Therefore, the overall space complexity of PDFlow is $O(|\mathcal{I}|d + d^2)$, indicating that memory usage is dominated by the number of items $|\mathcal{I}|$.

5 Experiments

This section reports our empirical study of PDFlow on four benchmark datasets. Our evaluation considers both ranking quality and popularity debiasing, and the key evaluation questions are summarized below.

- **RQ1**: How does PDFlow compare with other discriminative and generative baselines?
- **RQ2**: What happens when the residual popularity field or the alignment objective is removed from PDFlow?
- **RQ3**: How do the key hyperparameters (α, β, and λ) affect PDFlow's performance?
- **RQ4**: Does PDFlow increase the Hit Rate of unpopular items and reduce separation of embeddings driven by popularity?

Dataset	#Users	#Items	#Interactions	Mean length
Beauty	22,363	12,101	198,502	8.88
Toys	19,412	11,924	167,597	8.63
ML-100K	943	1,682	100,000	106.04
LastFM	1,090	3,646	52,551	48.21

Table 1: Dataset statistics.

5.1 Experimental Settings

Datasets. We evaluate PDFlow on four widely used benchmark datasets for sequential recommendation: **ML-100K** [Harper and Konstan, 2015], **Amazon Beauty** [Ni *et al.*, 2019], **Amazon Toys** [Ni *et al.*, 2019], and **LastFM** [Celma, 2010], covering diverse domains and varying sparsity levels. All datasets are treated as implicit feedback, where each user's interaction history is chronologically ordered to form behavior sequences. These datasets exhibit severe sparsity and long-tail item distributions, which tend to amplify popularity bias in recommendation. Dataset statistics are summarized in Table 1. Following common practice [Kang and McAuley, 2018], we adopt the leave-one-out evaluation protocol: for each user, the last interaction is used for testing, the second last for validation, and the remaining interactions for training.

Baselines. We compare PDFlow with representative baselines spanning different paradigms:

- **RNN-based**: GRU4Rec [Hidasi *et al.*, 2016].
- **Attention / Transformer-based**: SASRec [Kang and McAuley, 2018], BERT4Rec [Sun *et al.*, 2019], STOSA [Fan *et al.*, 2022], and DIN [Zhou *et al.*, 2018].
- **Contrastive learning-based**: DuoRec [Qiu *et al.*, 2022] and CL4SRec [Xie *et al.*, 2022].
- **Generative**: DreamRec [Yang *et al.*, 2023b], DiffuRec [Li *et al.*, 2024b], and DiQDiff [Mao *et al.*, 2025].

We implement baselines using official or widely adopted open-source codebases, follow the recommended settings, and perform validation-based tuning when applicable.

Implementation Details. PDFlow is implemented in PyTorch. Unless otherwise specified, the embedding dimension is set to 128 and the maximum sequence length is fixed to 50 for all datasets. We use the Adam optimizer with a batch size of 512, and the learning rate is set to 1×10^{-3}. For time-step sampling, we use mode sampling with heavy tails: we sample $k \sim \mathcal{U}(0, 1)$ and set the scale $s = 1.0$. For ODE-based sampling at inference, we use $N = 30$ Euler steps. All models are

Dataset	Metric	GRU4Rec	SASRec	DIN	BERT4Rec	STOSA	DuoRec	CL4SRec	DreamRec	DiffuRec	DiQDiff	PDFlow	Improv.(%)
ML-100k	HR@5	7.74	6.57	4.67	5.09	8.01	6.57	4.24	7.32	7.85	7.95	**9.63**	20.22
	HR@10	12.20	13.57	8.17	9.33	13.65	12.30	6.89	12.52	12.85	13.79	**14.5**	5.15
	HR@20	21.85	22.69	15.38	16.86	21.78	21.42	11.45	20.9	19.41	21.63	**23.52**	3.66
	NDCG@5	4.60	4.13	2.58	3.09	4.97	4.13	2.60	4.23	4.70	5.27	**5.8**	10.06
	NDCG@10	6.03	6.34	3.68	4.46	5.22	5.96	3.47	5.95	6.41	7.08	**7.35**	3.81
	NDCG@20	8.47	8.63	5.49	6.34	8.33	8.23	4.61	7.89	8.05	9.06	**9.59**	5.85
Beauty	HR@5	1.01	3.23	4.86	2.06	3.79	5.34	5.25	5.34	5.41	5.38	**5.94**	9.78
	HR@10	1.94	6.21	6.69	3.81	6.10	7.71	7.29	7.29	7.74	7.25	**8.70**	12.40
	HR@20	3.85	8.99	8.64	5.87	9.26	10.78	10.58	10.73	11.01	10.05	**12.25**	11.34
	NDCG@5	0.61	2.43	3.56	1.29	2.46	3.26	3.03	3.29	3.86	3.89	**4.08**	4.75
	NDCG@10	0.90	3.28	4.11	1.96	3.18	4.27	3.99	4.25	4.61	4.50	**4.97**	7.79
	NDCG@20	1.38	3.75	4.57	2.44	3.87	5.02	4.85	5.07	5.43	5.20	**5.86**	8.01
Toys	HR@5	1.10	4.59	5.51	1.91	4.41	5.64	5.48	5.49	5.67	5.51	**6.24**	10.19
	HR@10	1.85	6.67	6.78	2.97	7.08	7.25	6.87	7.08	7.48	7.26	**8.57**	14.61
	HR@20	3.18	9.42	8.61	4.68	9.83	9.86	10.09	9.75	9.85	10.11	**11.70**	15.73
	NDCG@5	0.70	3.14	4.01	1.14	3.31	3.17	3.34	4.05	4.27	4.23	**4.56**	6.83
	NDCG@10	0.94	3.82	4.47	1.51	4.01	3.98	4.27	4.61	4.85	4.83	**5.31**	9.54
	NDCG@20	1.27	4.73	5.03	1.95	4.46	4.78	5.08	5.26	5.45	5.42	**6.10**	12.04
LastFM	HR@5	3.12	4.13	4.18	2.94	3.57	4.50	4.30	4.42	4.54	4.45	**5.06**	11.67
	HR@10	4.04	6.33	5.78	4.59	5.98	6.90	6.35	6.07	6.24	6.63	**7.64**	10.68
	HR@20	5.41	9.27	9.09	5.96	8.67	10.94	9.04	9.83	10.12	9.67	**12.02**	9.84
	NDCG@5	2.17	2.84	2.77	1.98	3.14	3.27	2.45	3.21	3.37	3.18	**3.49**	3.39
	NDCG@10	2.45	3.55	3.37	2.52	3.87	4.14	3.11	3.99	4.03	4.16	**4.31**	3.56
	NDCG@20	2.80	4.29	4.19	2.86	4.26	4.81	3.78	4.89	4.96	4.87	**5.41**	9.08

Table 2: Overall performance comparison across four datasets. We report HR@K and NDCG@K (for $K \in \{5, 10, 20\}$). The best results are in **bold** and the second best results are underlined. All values are rounded to two decimals. Improv. is computed using unrounded values.

Dataset	Model Variant	HR@5	HR@10	HR@20	NDCG@5	NDCG@10	NDCG@20
Beauty	w/o $\mathcal{L}_{\text{align}}$	5.85	8.40	11.85	**4.14**	**4.97**	5.83
	w/o $\mathbf{v}_\phi^{\text{pop}}$	5.93	8.50	12.05	4.04	4.87	5.76
	PDFlow	**5.94**	**8.70**	**12.25**	4.08	**4.97**	**5.86**
Toys	w/o $\mathcal{L}_{\text{align}}$	6.20	8.44	11.41	4.54	5.26	6.01
	w/o $\mathbf{v}_\phi^{\text{pop}}$	6.18	**8.64**	11.57	**4.56**	**5.35**	6.09
	PDFlow	**6.24**	8.57	**11.70**	**4.56**	5.31	**6.10**
ML-100k	w/o $\mathcal{L}_{\text{align}}$	7.70	14.03	22.39	5.18	7.18	9.28
	w/o $\mathbf{v}_\phi^{\text{pop}}$	7.93	13.94	21.82	5.05	6.99	8.96
	PDFlow	**9.63**	**14.5**	**23.52**	**5.8**	**7.35**	**9.59**
LastFM	w/o $\mathcal{L}_{\text{align}}$	4.82	7.26	11.01	2.87	3.64	4.55
	w/o $\mathbf{v}_\phi^{\text{pop}}$	3.61	6.12	10.29	2.74	3.56	4.64
	PDFlow	**5.06**	**7.64**	**12.02**	**3.49**	**4.31**	**5.41**

Table 3: Ablation study summary across datasets. Best results are in **bold** and second-best are underlined.

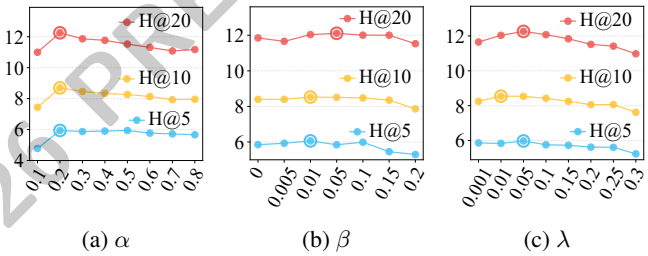


Figure 3: **Hyperparameter sensitivity on Amazon Beauty (HR@K).** We vary α (loss balance), β (popularity regularization), and λ (debiasing strength), and report HR@K.

trained for up to 500 epochs and evaluated every 20 epochs; early stopping is applied with a patience of 5 validations. We evaluate models and report HR@K and NDCG@K with $K \in \{5, 10, 20\}$. Following prior work [Kang and McAuley, 2018], we use the full item set as candidates for ranking when computing these metrics.

Reproducibility. All experiments are conducted on Ubuntu 22.04.1 LTS with PyTorch 2.1.2, NumPy 1.26.4, and SciPy 1.15.3. The server is equipped with an Intel(R) Xeon(R) Platinum 8474C CPU (48 cores) and two NVIDIA RTX 4090 GPUs. Unless stated otherwise, all experiments are conducted using a single GPU for each run.

5.2 Overall Performance (RQ1)

Table 2 reports the overall results on four benchmark datasets. Across datasets and evaluation cutoffs, PDFlow achieves the best or competitive performance in most settings. In particular, PDFlow consistently outperforms diffusion-based generators such as DiffuRec and DiQDiff on all datasets, indicating that incorporating popularity-aware correction is beneficial beyond standard generative dynamics.

The improvements are especially pronounced on the two Amazon datasets, where user–item interactions follow highly long-tailed distributions. For example, PDFlow improves HR@20 by up to 11.34% on Beauty and 15.73% on Toys. These gains suggest that the proposed method is particularly effective in scenarios where iterative generation is more likely to over-concentrate on popular items.

On ML-100K, where sequences are longer and the interaction data is relatively denser, PDFlow still yields consistent improvements across all cutoffs (e.g., 5.85% on NDCG@20). This indicates that the benefit of popularity-aware correction is not limited to extremely sparse settings. On LastFM, the improvements are smaller but remain consistent across metrics, reflecting domain-dependent differences in popularity concentration and recommendation difficulty. Overall, PDFlow maintains strong ranking accuracy across various datasets while reducing popularity-driven effects.

5.3 Ablation Study (RQ2)

In order to verify the effectiveness of major individual components of PDFlow, we performed an ablation study on four

datasets. Our variants are as follows: (1) **w/o** $\mathcal{L}_{\text{align}}$, which removes the cross popularity alignment loss; (2) **w/o** $\mathbf{v}_\phi^{\text{pop}}$, which disables the residual popularity subtraction in both training and inference by setting $\lambda = 0$ during the entire process; (3) **PDFlow**, the full model.

Table 3 shows that removing either component leads to lower ranking performance across almost all datasets and cutoffs. Rather than a one-sided effect, both ablations (w/o $\mathcal{L}_{\text{align}}$ and w/o $\mathbf{v}_\phi^{\text{pop}}$) consistently underperform the full model. This indicates that each module of our PDFlow contributes meaningfully to the final performance. The magnitude of degradation, however, is dataset dependent. On Amazon Beauty and Amazon Toys, the drops are present but relatively modest, and the two ablated variants often stay close to each other across HR@K and NDCG@K. In contrast, the gap becomes much more visible on LastFM, where removing either component results in a clear decrease on both HR and NDCG, and the full model keeps a noticeable margin across multiple cutoffs. A particularly striking case is ML-100K at HR@5: the full model achieves 9.63, while removing the popularity field yields 7.93, meaning that PDFlow improves over w/o $\mathbf{v}_\phi^{\text{pop}}$ by about 21.44%. This suggests that the two modules not only act as fine-grained regularizers, but also can materially affect the early top-K ranking behavior in some regimes. Overall, the ablation results support a simple conclusion: both $\mathbf{v}_\phi^{\text{pop}}$ and $\mathcal{L}_{\text{align}}$ are needed to recover the strongest and most stable performance across datasets with different popularity and sequence characteristics.

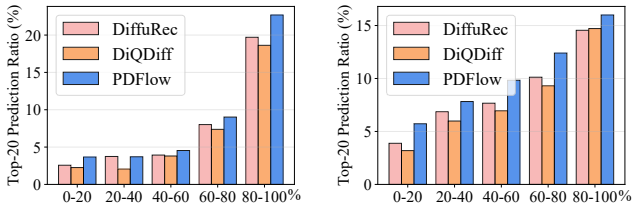
5.4 Impact of Hyperparameter Settings (RQ3)

To evaluate the sensitivity of the three key parameters, loss balancing weight α, popularity regularization β, and debiasing strength λ, we conduct tests on the Amazon Beauty dataset as shown in Figure 3. PDFlow maintains stable HR@K and consistently outperforms baselines, demonstrating strong robustness. Specifically, performance peaks at $\alpha = 0.2$ and $\lambda = 0.05$, showing that moderate auxiliary supervision and debiasing are essential to rectify trajectory drift without sacrificing mainstream interest modeling. The model is robust to β, only declining slightly when $\beta > 0.1$, suggesting that excessive regularization may suppress useful interest signals.

5.5 Popularity Bias Analysis (RQ4)

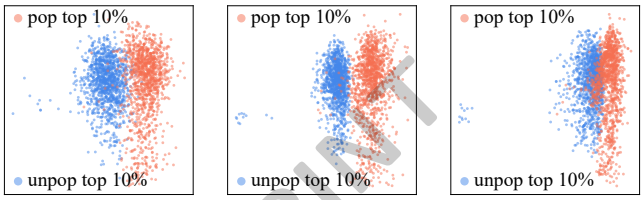
We analyze the popularity-binned distribution of Top-20 recommendations. Items are partitioned into five bins from high to low popularity; lower-popularity bins largely correspond to long-tail items under the long-tailed interaction distribution. As shown in Figure 4, PDFlow allocates a larger share of recommendations to low-popularity bins than diffusion-based baselines, while maintaining strong overall ranking performance (Section 5.2).

To further inspect whether this difference is reflected at the model's representation level, we visualize item embeddings on Amazon Beauty by marking popular (top 10%) and unpopular (bottom 10%) items with different colors. As shown in Figure 5, DiffuRec and DiQDiff exhibit a clear separation between the two groups. Under PDFlow, the two groups appear closer and more overlapping, indicating a weaker stratification between popular and long-tail items in the embedding space.



(a) Amazon Beauty (b) Amazon Toys

Figure 4: **Popularity-binned prediction ratio in Top-20.** Items are partitioned into five bins from high to low popularity.



(a) DiffuRec (b) DiQDiff (c) PDFlow

Figure 5: **PCA visualization of popular and unpopular items on Amazon Beauty.** We select the top 10% and bottom 10% items by interaction frequency (popularity), project item embeddings to 2D via PCA, and align the centroid direction from popular to unpopular items for consistent visual comparison.

Taken together, these observations suggest that PDFlow not only redistributes recommendation exposure toward less popular items, but also reduces popularity-induced separation in the learned representations, which can help alleviate bias during rounding-based retrieval.

6 Conclusion

In this paper, we propose PDFlow to address the popularity bias in generative sequence recommendation. PDFlow constructs a parameterized vector field to mitigate bias, comprising a mainstream interest field and a residual popularity field that eliminate bias during both training and inference. Concurrently, it introduces a cross-popularity alignment mechanism to narrow the representational gap between popular and long-tail items. Experiments demonstrate that PDFlow effectively reduces popularity bias while improving accuracy. Future work will focus on adaptive bias reduction and extending the framework to address other bias scenarios.

Acknowledgments

This research was partially supported by the NSFC (62376180), the National Key Research and Development Program of China(2023YFF0725002), Suzhou Science and Technology Development Program(SYG202328) and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

Contribution Statement

Zican Yang and Zeyu Li contributed equally to this work.

References

[Abdollahpouri *et al.*, 2017] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *RecSys*, pages 42–46, 2017.

[Abdollahpouri *et al.*, 2019] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. *arXiv preprint arXiv:1901.07555*, 2019.

[Celma, 2010] Òscar Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[Cui *et al.*, 2019] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, pages 9268–9277, 2019.

[Esser *et al.*, 2024] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

[Fan *et al.*, 2022] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S Yu. Sequential recommendation via stochastic self-attention. In *WWW*, pages 2036–2047, 2022.

[Harper and Konstan, 2015] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2015.

[Hidasi *et al.*, 2015] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[Hidasi *et al.*, 2016] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[Joachims *et al.*, 2017] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *WSDM*, pages 781–789, 2017.

[Kang and McAuley, 2018] Julian Kang and Julian McAuley. Self-attentive sequential recommendation. In *ICDM*, pages 197–206, 2018.

[Kang *et al.*, 2019] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.

[Li *et al.*, 2020] Jiacheng Li, Pengjie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In *WSDM*, pages 322–330, 2020.

[Li *et al.*, 2024a] Wu Li, Rui Huang, Haijun Zhao, Chi Liu, Kai Zheng, Qi Liu, et al. Dimerec: A unified framework for enhanced sequential recommendation via generative diffusion models. In *WSDM*, 2024.

[Li *et al.*, 2024b] Zihao Li, Aixin Sun, and Chenliang Li. DiffuRec: A diffusion model for sequential recommendation. *ACM Transactions on Information Systems*, 42(3):66:1–66:28, 2024.

[Lipman *et al.*, 2023] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

[Liu *et al.*, 2023] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[Liu *et al.*, 2025] Feng Liu, Lixin Zou, Xiangyu Zhao, Min Tang, Liming Dong, Dan Luo, Xiangyang Luo, and Chenliang Li. Flow matching based sequential recommender model. *arXiv preprint arXiv:2505.16298*, 2025.

[Liu *et al.*, 2026] Lingfeng Liu, Yixin Song, Dazhong Shen, Bing Yin, Hao Li, Yanyong Zhang, and Chao Wang. Rethinking popularity bias in collaborative filtering via analytical vector decomposition. In *KDD*, 2026.

[Mao *et al.*, 2025] Wenyu Mao, Shuchang Liu, Haoyang Liu, Haozhe Liu, Xiang Li, and Lantao Hu. Distinguished quantized guidance for diffusion-based sequence recommendation. In *WWW*, 2025.

[Ni *et al.*, 2019] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*, pages 188–197, 2019. Amazon Beauty Dataset.

[Ning *et al.*, 2023] Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. *arXiv preprint arXiv:2308.15321*, 2023.

[Ning *et al.*, 2024] Wentao Ning, Reynold Cheng, Xiao Yan, Ben Kao, Nan Huo, Nur Al Hasan Haldar, and Bo Tang. Debiasing recommendation with personal popularity. In *WWW*, 2024.

[Qin *et al.*, 2023] Yiming Qin, Huangjie Zheng, Jiangchao Yao, Mingyuan Zhou, and Ya Zhang. Class-balancing diffusion models. In *CVPR*, pages 18434–18443, 2023.

[Qiu *et al.*, 2022] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. Contrastive learning for representation degeneration problem in sequential recommendation. In *WSDM*, pages 813–823, 2022.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820, 2010.

[Schnabel *et al.*, 2016] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandar, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *ICML*, pages 1670–1679, 2016.

[Sun *et al.*, 2019] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pages 1441–1450, 2019.

[Tan *et al.*, 2025] Shiyin Tan, Dongyuan Li, Renhe Jiang, Zhen Wang, Xingtong Yu, and Manabu Okumura. Taming recommendation bias with causal intervention on evolving personal popularity. In *KDD*, 2025.

[Tang and Wang, 2018] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573, 2018.

[Wang *et al.*, 2023] Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. In *SIGIR*, 2023.

[Wei *et al.*, 2021] 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 *KDD*, pages 1791–1800, 2021.

[Xie *et al.*, 2021] Zhe Xie, Chengxuan Liu, Yichi Zhang, Hongtao Lu, Dong Wang, and Yue Ding. Adversarial and contrastive variational autoencoder for sequential recommendation. In *WWW*, 2021.

[Xie *et al.*, 2022] Xu Xie, Fei Sun, Zaidong Liu, Shiwen Wu, Jinyun Lao, Thanard Kurutach, Zhwang Ji, Kaituo Chen, Fan Cheng, and Shuaicheng Yang. Contrastive learning for sequential recommendation. In *SIGIR*, 2022.

[Yang *et al.*, 2023a] Yuhao Yang, Chao Huang, Lianghao Xia, Chunzhen Huang, Da Luo, and Kangyi Lin. Debiased contrastive learning for sequential recommendation. In *WWW*, 2023.

[Yang *et al.*, 2023b] Zhengyi Yang, Jiancan Wu, Zhicai Wang, Yancheng Yuan, Xiang Wang, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. In *NeurIPS*, 2023.

[Ye *et al.*, 2025] Xiaoxin Ye, Chengkai Huang, Hongtao Huang, and Lina Yao. Gaussian mixture flow matching with domain alignment for multi-domain sequential recommendation. *arXiv preprint arXiv:2510.21021*, 2025.

[Yuan *et al.*, 2019] Fajie Yuan, Alexandros Karatzoglou, Ioannis Engelhardt, Chenyang Guo, and Jian Liao. A simple convolutional generative network for next item recommendation. In *WSDM*, pages 582–590, 2019.

[Zhang *et al.*, 2021] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. Causal intervention for leveraging popularity bias in recommendation. In *SIGIR*, pages 11–20, 2021.

[Zhao *et al.*, 2023] 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 (TKDE)*, 35(10):9920–9931, 2023.

[Zheng *et al.*, 2021] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. Disentangling user interest and conformity for recommendation with causal embedding. In *WWW*, pages 2980–2991, 2021.

[Zhou *et al.*, 2018] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *KDD*, pages 1059–1068, 2018.