Improving Long-tail User CTR Prediction via Hierarchical Distribution Alignment

Yifan Wang
DCST, Tsinghua University
Beijing, China
yf-wang21@mails.tsinghua.edu.cn

Weizhi Ma*
AIR, Tsinghua University
Beijing, China
mawz@tsinghua.edu.cn

Min Zhang*
DCST, Tsinghua University
Beijing, China
z-m@tsinghua.edu.cn

Xiaoxiao Xu
Kuaishou Technology
Beijing, China
xuxiaoxiao05@kuaishou.com

Zhiqiang Liu
Kuaishou Technology
Beijing, China
zhiqliu1103@gmail.com

Shaoping Ma
DCST, Tsinghua University
Beijing, China
msp@tsinghua.edu.cn

Abstract

Click-Through Rate (CTR) prediction is a fundamental task in online advertising and recommender systems, requiring the effective modeling of feature interactions. While existing methods have improved overall prediction performance, the performance of long-tail users with limited historical data remains suboptimal. These users face two primary challenges: (i) insufficient training data leading to inaccurate predictions, and (ii) an imbalanced sample distribution that biases model learning toward head users. To address these challenges, we propose a novel framework that enhances long-tail user performance through hierarchical distribution alignment, hierarchical residual learning, and adaptive distrbution calibration. Our method first captures the shared patterns between head and long-tail users via hierarchical distribution alignment, then learns group-specific information through hierarchical residual learning. Additionally, we re-balance the sample distributions of head users and long-tail users by dynamic reweighting. To counteract potential biases introduced by reweighting, we further incorporate a distribution calibration module. Our method is model-agnostic and can be seamlessly integrated into various CTR prediction architectures that rely on feature interactions. Extensive experiments on public datasets and an online experiment demonstrate that our approach significantly improves accuracy and fairness for long-tail users while maintaining similar or even better overall performance.[1]

CCS Concepts

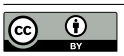
• **Information systems** → **Recommender systems**.

Keywords

CTR prediction; long-tail users; distribution alignment

*Corresponding authors

[1]This work is supported by the Natural Science Foundation of China (Grant No.U21B2026, 62372260) and Kuaishou.



KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737003

ACM Reference Format:
Yifan Wang, Weizhi Ma, Min Zhang*, Xiaoxiao Xu, Zhiqiang Liu, and Shaoping Ma. 2025. Improving Long-tail User CTR Prediction via Hierarchical Distribution Alignment. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737003

KDD Availability Link:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.15562175.

1 Introduction

Predicting the probability of users clicking on ads or items is critical to many online applications, such as digital display advertising and recommender systems, which are also known as the Click-Through Rate (CTR) prediction tasks [2, 7]. Since platforms often have abundant user and item features (e.g., gender for users and categories for items) available for training CTR models, learning effective feature interactions has become a critical part of the CTR prediction task and has attracted widespread attention from both academia and industry in recent years [2, 7, 14, 34]. It can be classified into implicit feature interaction and explicit feature interaction [23]. Implicit feature interaction aims to automatically learn complex non-manually defined data patterns and high-order feature interactions using deep neural networks, while explicit feature interaction aims to model the combinations and relationships between input features directly through predefined functions. These two types of feature interactions are both important for CTR prediction, and current CTR prediction methods typically include both modules [2, 7, 23].

Focusing on improving the overall system performance can potentially lead to the neglect of certain disadvantaged groups (e.g., long-tail users) [15, 16, 24, 25, 33]. In recent years, increasing attention has been given to subgroup performances in recommendation systems [3, 4, 10, 11]. However, we observe that most studies focus on top-N recommendation tasks based on ID information, overlooking CTR prediction task that rely on rich feature interactions. As shown in Figure 1, a long-tail phenomenon also exists on the user side in CTR datasets, where the top 20% of users may account for around 50% of the training data. Meanwhile, long-tail users receive a poorer recommendation performance.

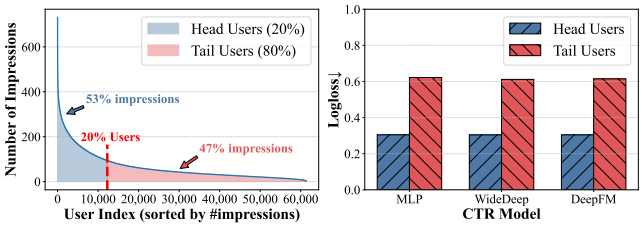


Figure 1: The user distribution (left) and the performance gap (right) between tail and head users on the Taobao dataset.

Improving the performance of long-tail users for CTR prediction is also a crucial issue. On the one hand, long-tail users are often more prone to churn, and optimizing their performance contributes to the platform's long-term development. On the other hand, enhancing the performance of long-tail users often aligns with the requirements of fairness among user groups [17, 29, 31]. However, previous methods proposed for long-tail users are often re-ranking approaches targeting top-N ranking tasks or ID-based collaborative filtering methods, making them difficult to apply to CTR prediction tasks that involve rich feature interactions.

We consider two main challenges faced by long-tail users that lead to their poor performance: the first challenge is that long-tail users inherently have less informative data, making it difficult for the model to make accurate predictions. The second challenge is that, compared to head users, the sample proportion of long-tail users is often smaller, which may cause the model parameters to prioritize learning from head users. For the first challenge, we leverage the shared knowledge between head users and long-tail users to assist the learning process for long-tail users. Specifically, we first learn the shared knowledge through hierarchical distribution alignment. In detail, inspired by domain adaptation studies [5], we consider head and long-tail users as different domains and align their representation distributions. The aligned representations, invariant across head and long-tail users, serve as the shared knowledge. After obtaining the shared knowledge, we further learn the specific knowledge unique to head users and long-tail users based on this shared foundation. For the second challenge, we re-balance the distributions of head users and long-tail users by dynamic reweighting. However, reweighting may distort the original training data distribution, leading to a systematic bias in predicted probabilities. To address this issue, we introduce a distribution calibration module to separately calibrate the distributions of head users and long-tail users, thereby mitigating the distribution shift caused by reweighting. To demonstrate the effectiveness, we conduct extensive experiments on two public datasets. Experimental results show that our method can effectively improve long-tail user performance in CTR prediction. Our main contributions can be summarized as follows.

- To the best of our knowledge, this is the first work that attempts to improve the performance of long-tail users in the CTR prediction task from the perspective of distribution alignment.
- We theoretically analyze the shortcoming of existing distribution alignment methods when applied to this problem and further proposed a hierarchical distribution alignment framework to improve the performance of long-tail users.
- Comparative experimental results on two public datasets and an online experiment demonstrate the effectiveness of our proposed framework.

2 Related Work

2.1 Long-tail Problem in Recommendation

The long-tail problem in recommendation can be considered separately from the user and item perspectives. Among these, the item-side long-tail problem has attracted more attention [26, 28, 35, 36]. It is important to note that the long-tail problem is distinct from the cold-start problem, as the bottom 80% of long-tail users and items may still have a significant amount of training data and are not necessarily cold-start users or items.

For the item-side long-tail problem, existing work can be roughly categorized into three types: (i) causal inference-based methods [26, 36]; (ii) content-enhanced methods [18]; (iii) knowledge transfer-based methods[28, 35]; Due to the asymmetry in the roles of users and items, these methods cannot directly apply to the user side.

For the user-side long-tail problem, existing studies often focus on Top-N recommendation tasks and can be roughly divided into three categories: (i) data augmentation-based methods [1]: these methods generate virtual interactions for long-tail users through resampling or more sophisticated data generation techniques. (ii) knowledge transfer-based methods [8, 9]: these approaches enhance the collaborative signals and enrich the representations of long-tail users by identifying and leveraging similar users. (iii) re-ranking-based methods [16]: these methods adjust the ranking lists during the re-ranking phase to improve the recommendation performance for long-tail users.

However, these methods do not involve feature interactions, making them difficult to directly apply to CTR prediction tasks that involve rich feature interactions.

2.2 User Fairness in Recommendation

Poor long-tail user performance is also a user unfairness problem. Current research on user fairness can be roughly divided into two groups: learning fair user representations [17, 29] and producing fair recommendation outcomes [4, 10, 15, 16, 31, 33]. The former is related to process fairness, while the latter focuses on the fairness of recommendation performance received by different users, which has attracted more attention as it is more related to user satisfaction. These fairness methods on outcome fairness can be roughly grouped into three categories: (i) fairness regularization[10, 15, 33]. (ii) distributionally robust optimization[27]. (iii) re-ranking[4, 16].

Unlike the previous work, we focus on CTR prediction tasks that involve feature interactions and design a hierarchical distribution alignment framework applicable to various CTR prediction models to enhance the performance of long-tail users.

2.3 Distribution Alignment

In this work, we employ distribution alignment techniques to learn the shared knowledge between long-tail and head users. Distribution alignment techniques have been widely applied in various tasks, such as domain adaptation[5], image generation[6],

and fair representation learning[29]. Existing distribution alignment methods can be roughly categorized into the following types: (i) adversarial learning-based methods[5, 29]: These methods employ an additional discriminator to distinguish between different distributions and perform adversarial training to align them. (ii) regularization-based methods [21, 30]: These approaches directly compute the difference between distributions (e.g., using maximum mean discrepancy [30] or Wasserstein distance [21]) and incorporate the difference term into the loss function for optimization. (iii) reweighting-based methods[19]: These methods align distributions through reweighting techniques, such as importance reweighting, to adjust the distribution weights.

However, we find that directly aligning representation distributions may fail to learn invariant representations. To address this problem, we further design a hierarchical distribution alignment method tailored for CTR prediction tasks.

3 Preliminary

3.1 CTR Prediction

In advertising and recommendation platforms, the user behaviors for the display ads or exposed items are logged to train the CTR prediction model. It can be typically considered a binary classification task that utilizes user profiles, item attributes and context as features to predict the probability of a user clicking on an exposed item. Without loss of generalizability, we assume all the features are categorical as numerical features can be converted into categorical features via bucketization. Further, we can define a CTR input sample as $(x_1, x_2, ..., x_n, y)$, where x_i denotes the i-th categorical feature, and $y \in \{0, 1\}$ is the ground truth for user click behavior.

A CTR prediction model aims to predict y and then rank items based on the predicted probabilities $\hat{y}$ and other information such as ad price and predicted conversion rate. Note that CTR prediction requires an accurate and unbiased prediction of probabilities as the prediction may be combined with other information, thus the log loss is a very important metrics for CTR prediction, which is different from traditional classification tasks mainly focusing on the classification accuracy.

Feature interaction is a key component for CTR prediction methods, which can be classified into implicit feature interaction and explicit feature interaction [23]. Implicit feature interaction aims to automatically learn complex non-manually defined data patterns and high-order feature interactions using deep neural networks. Explicit feature interaction aims to model the combinations and relationships between input features directly through predefined functions, such as the Hadamard Product. These two types of feature interactions are both important for CTR prediction, and current CTR prediction methods typically include both implicit and explicit feature interaction modules [2, 7].

3.2 Objectives for Tail User CTR Prediction

Based on the number of CTR training samples logged, users can be divided into long-tail (inactive) users and head (active) users. Following previous work [16], we consider the top 20% active users with the most samples as head users, and the left 80% users as long-tail users. Formally, let the long-tail users $\mathcal{U}_L = \{u_1, u_2, ..., u_j\}$, the head users $\mathcal{U}_H = \{u_{j+1}, ..., u_n\}$, the predicted and ground truth information for the i-th user is $\mathcal{Y}_i = \{(\hat{y}_1, y_1), ..., (\hat{y}_m, y_m)\}$, where m is the sample number for the i-th user. Further, we can define the user group performance in CTR prediction:

Definition 3.1 (User Group Performance in CTR prediction). Given a user group $\mathcal{U}$, and the corresponding prediction information $\{\mathcal{Y}_i\}_{i=1}^{|\mathcal{U}|}$, the group performance for $\mathcal{U}$ can be defined as:

$$\mathrm{GP}(\mathcal{U}) = \frac{1}{|\mathcal{U}|} \sum_{i=1}^{|\mathcal{U}|} \mathcal{M}(\mathcal{Y}_i) \tag{1}$$

Here $\mathcal{M}(\cdot)$ is a user-level metrics for CTR prediction. In this work, we use the average log loss as $\mathcal{M}(\cdot)$.

Based on the definition, we can improve the long-tail user performance from two perspective: accuracy and fairness. For accuracy, we naturally tends to improve the performance for long-tail users. For fairness, following previous work [16], we focus on the performance gap between long-tail users and head users.

In summary, we have three optimization objectives:

- **Overall Accuracy**: we aim to improve the accuracy for all users in CTR prediction.
- **Long-tail Users Accuracy**: we particularly aim to improve the accuracy of long-tail users, i.e., $\mathrm{GP}(\mathcal{U}_L)$.
- **Long-tail Users Fairness**: we aim to improve the fairness between long-tail and head users, i.e., decrease the performance gap $|\mathrm{GP}(\mathcal{U}_L) - \mathrm{GP}(\mathcal{U}_H)|$.

4 Method

4.1 Overview

We consider two main challenges faced by long-tail users that lead to their poor performance: the first challenge is that long-tail users inherently have less informative data, making it difficult for the model to make accurate predictions. The second challenge is that, compared to head users, the sample proportion of long-tail users is often smaller, which may cause the model parameters to prioritize learning from head users.

For the first challenge, we leverage the shared knowledge between head users and long-tail users to assist the learning process for long-tail users (**Stage 1** and **Stage 2** in Figure 2). Specifically, we first learn the shared knowledge through hierarchical distribution alignment. In detail, we treat head users and long-tail users as different domains. Inspired by domain adaptation studies [5], we align the representation distributions of head and long-tail users. The aligned representations, invariant across head and long-tail users, serve as the shared knowledge. After obtaining the shared knowledge, we further learn the specific knowledge unique to head users and long-tail users based on this shared foundation.

For the second challenge, we re-balance the distributions of head users and long-tail users via dynamic reweighting. However, reweighting may distort the original training data distribution, leading to a systematic bias in predicted probabilities. To address this issue, we introduce a distribution calibration module (**Stage 3** in in Figure 2) to separately calibrate the distributions of head users and long-tail users, thereby mitigating the distribution shift caused by reweighting.

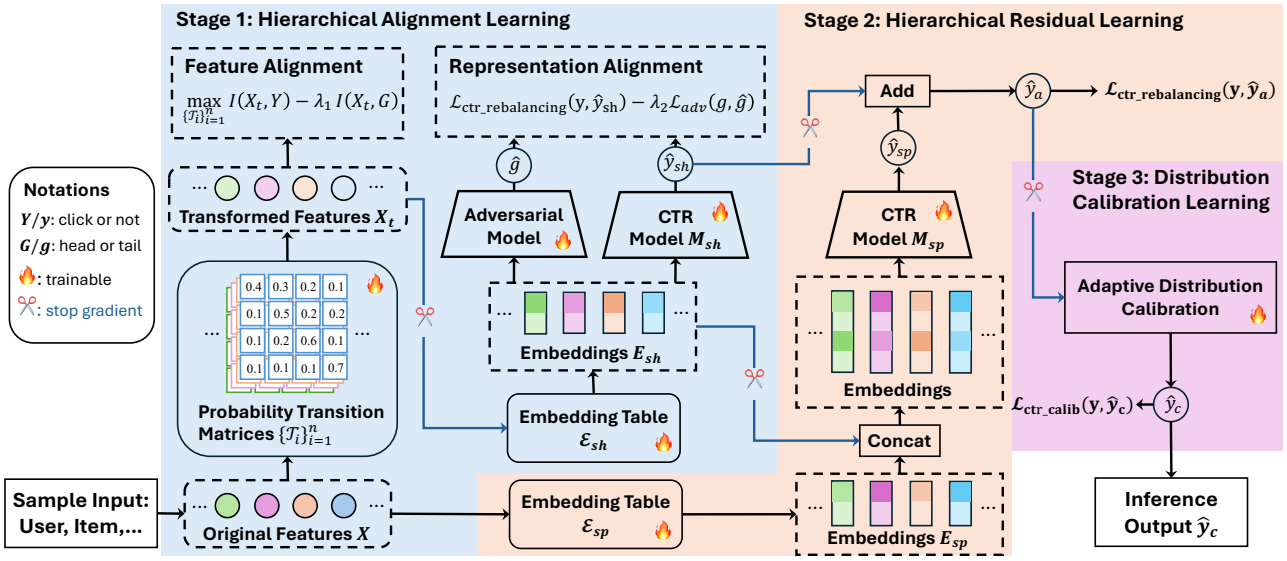


Figure 2: Illustration for our framework.

4.2 Hierarchical Distribution Alignment

Drawing inspiration from domain adaptation studies [5], we treat long-tail users and head users as different domains and learn the shared knowledge between them through distribution alignment. Formally, given the input feature $X = [x_1, ..., x_n]$, label Y, and group indicator G, we aim to learn a transformation function $\mathcal{K}(\cdot)$ to obtain shared sample representations $R_{sh} = \mathcal{K}(X)$, satisfying $P(R_{sh}|G) = P(R_{sh})$.

A natural idea is to use adversarial learning for representation distribution alignment, but it may fail to learn the ideal transformation function (the ideal properties of the transformation function are discussed in Section 4.2.1, and the corresponding theoretical analysis is in Section 4.2.4). Therefore, we propose hierarchical distribution alignment, which combines distribution alignment at the **feature level** (Section 4.2.2) and the **representation level** (Section 4.2.3), achieving better distribution alignment results (Section 4.2.4).

4.2.1 Distribution Alignment Properties. The transformation function $\mathcal{K}(\cdot)$ should satisfy **three properties**:

- The transformation should filter out group-specific information as much as possible, i.e., minimize $\mathcal{I}(\mathcal{K}(X), G)$.
- Based on the first property, the transformation should preserve information relevant to the task labels as much as possible, i.e., maximize $\mathcal{I}(\mathcal{K}(X), Y)$.
- The transformation should be feature-decoupled. That means that $R_{sh} = [r_1, ..., r_n]$, where r_i is only determined by x_i.

The first and second properties are relatively straightforward, aiming to extract as much task-relevant knowledge as possible that is shared across different groups from the original feature. As for the third property, this is because we aim to design a model-agonistic framework. Since the explicit interaction modules commonly used in CTR models (e.g., DeepFM [7]) require the input features to be decoupled, we should ensure that the transformation also maintains feature decoupling.

4.2.2 Probabilistic Feature Mapping. First, we attempt to perform distribution alignment at the feature level. We introduce a feature-level probability mapping module $\mathcal{T} : X = [X_1, ..., X_n] \rightarrow X_t = [X_1^t, ..., X_n^t]$, which maps the original feature inputs to new feature inputs. To ensure that the mapping is feature-decoupled, we require each feature's mapping independent and unaffected by others, i.e., learn n probability mapping function $\{\mathcal{T}_i : X_i \rightarrow X_i^t\}_{i=1}^n$. Essentially, $\mathcal{T}_i$ is a probability transition matrix of size $p_i \times q_i$, where p_i is the number of unique values in the original feature, and q_i is the number of unique values in the transformed feature. For simplicity, we set $q_i = p_i$. According to the alignment properties, the optimization objective for the probability transition matrix is as follows:

$$\begin{aligned} \max_{\{\mathcal{T}_i\}_{i=1}^n} \quad & \mathcal{I}(X_t, Y) \\ \text{s.t.} \quad & \mathcal{I}(X_t, G) = 0 \end{aligned} \tag{2}$$

In terms of practical optimization, it is worth noting that both our features and group information are discretized, so mutual information can be directly calculated. However, the above optimization is a non-convex optimization problem under non-convex constraints, making it difficult to optimize directly. Therefore, we convert it into a soft form:

$$\max_{\{\mathcal{T}_i\}_{i=1}^n} \quad \mathcal{I}(X_t, Y) - \lambda_1 \cdot \mathcal{I}(X_t, G) \tag{3}$$

In practice, the joint probability space of multiple features can be very large and sparse . Therefore, we optimize the mutual information between a single feature, the task label, and group attributes at a time.

4.2.3 Adversarial Representation Learning. Next, we attempt to perform distribution alignment at the representation level. A natural and straightforward approach is adversarial learning [5]. Typically, given the transformed feature input $[x_1^t, x_2^t, ..., x_n^t]$, we first obtain multiple embeddings $[e_1, e_2, ..., e_n]$ via embedding layers $\mathcal{E}_{sh}$. We then introduce a filtering neural network $\mathcal{F}(\cdot)$ to extract the shared

representations E_{sh}:

$$E_{sh} = \mathcal{F}(Concat([e_1, e_2, ..., e_n])) \tag{4}$$

where $Concat(\cdot)$ is the concatenate function.

To learn shared knowledge about CTR prediction, we introduce a discriminator neural network $\mathcal{D}(\cdot)$, and the CTR prediction network $\mathcal{M}_{sh}(\cdot)$. The optimization of adversarial learning can be defined as the following minimax problem:

$$\min_{\mathcal{E}_{sh}, \mathcal{M}_{sh}, \mathcal{F}} \max_{\mathcal{D}} \mathcal{L}_{ctr} - \lambda_2 \cdot \mathcal{L}_{adv} \tag{5}$$

where $\mathcal{L}_{ctr}$ is the training loss for CTR prediction such as binary cross entropy, and $\mathcal{L}_{adv}$ is the cross entropy loss for the discriminator $\mathcal{D}(\cdot)$ to discover group information from filtered representations E_{sh}. It can be theoretically proved that when the discriminator loss $\mathcal{L}_{adv}$ achieves optimal, then the mutual information $\mathcal{I}(E_{sh}, G)$ is zero, i.e., $P(E_{sh}|G) = P(E_{sh})$ [6]. At the same time, according to the information bottleneck theory [20], minimizing $\mathcal{L}_{ctr}$ is equivalent to maximizing $\mathcal{I}(E_{sh}, Y)$. Thus, the aforementioned method can effectively satisfy properties 1 and 2.

Specifically, the above minimax problem is optimized through minimizing the following two losses alternatively:

$$\mathcal{L}_{adv} = -\sum_{i=1}^{n} \sum_{s:s=s_u} \log \mathcal{D}(E_{sh}) \tag{6}$$

$$\mathcal{L}_{align} = \mathcal{L}_{ctr} - \lambda_2 \cdot \mathcal{L}_{adv} \tag{7}$$

However, one problem of the above approach is that if we use a neural network as the filtering network, it violates the third property as the filtered representations E_{sh} have their feature information coupled together.

To mitigate this issue, we remove the filtering network and use an equivariant network as $\mathcal{F}(\cdot)$, directly learning embedding tables that contains only the shared knowledge:

$$E_{sh} = Concat([e_1, e_2, ..., e_n]) \tag{8}$$

The overall process of hierarchical alignment learning is as follows: we first learn the probability mapping modules $\{\mathcal{T}_i\}_{i=1}^n$. After obtaining the probability mapping matrix, we fix it and then perform adversarial learning according to Eq.(5). Upon completing the learning process, we acquire the probability mapping modules $\{\mathcal{T}_i\}_{i=1}^n$, the embedding table $\mathcal{E}_{sh}$ that preserves shared knowledge, and the CTR model $\mathcal{M}_{sh}$ for predictions based on the shared knowledge, with the corresponding prediction $\hat{y}_{sh}$.

4.2.4 Theoretical Analysis. In this section, we theoretically demonstrate that only performing adversarial learning at the representation level may not yield the desired invariant knowledge representations (Theorem 4.1). In addition, probabilistic feature mapping will not lead to a worse optimal solution (Theorem 4.2). Furthermore, our subsequent experiments (Section 5.5) have verified that probabilistic feature mapping can achieve better distribution alignment.

Theorem 4.1 (Discrete Optimization Limit). *Let $\mathcal{P}$ be any fixed data distribution and $G \in \{0, 1\}$ be a binary group attribute. If there exists a categorical feature X such that $X \not\perp G$, then there exists a constant $c > 0$, such that for any non-trivial embedding table parameters (i.e., not all embeddings are the same), the mutual information $\mathcal{I}(E_{sh}, G)$ for E_{sh} in the form of Eq.(8) has $\mathcal{I}(E_{sh}, G) > c$. This implies that it is impossible to learn a perfect shared representation.*

Remark. Note that Theorem 4.1 does not impose any constraints on how the embedding parameters are learned. This means that this theorem is not limited to the adversarial distribution alignment method used in our method, but is also applicable to other distribution alignment methods [21, 30].

Theorem 4.2 (improvement property). *Adding the probabilistic feature mapping into optimization through Eq.(3) before the adversarial learning in Eq.(5) does not lead to an increase in the minimum value of $\mathcal{I}(E_{sh}, G)$, and there exists a case where the minimum value is reduced to 0 after adding probabilistic feature mapping.*

Due to space limitations, the proof can be found in the appendix.

4.3 Hierarchical Residual Learning

We have learned shared knowledge between long-tail users and head users through hierarchical distribution alignment. However, there may still be unique information specific to long-tail and head users. Ignoring this information could lead to suboptimal performance. Therefore, after the first-stage alignment learning, we perform the second-stage residual learning to capture this residual information.

Specifically, we conduct hierarchical residual learning from two perspectives: **representation level** and **prediction level**. We input the original features X into a new embedding table $\mathcal{E}_{sp}$ to obtain the specific representations for the corresponding samples.

At the representation level, we aggregate them with the fixed shared representations:

$$e_i = Concat([e_i^{sh}, e_i^{sp}]) \tag{9}$$

Based on this, we obtain prediction scores through a new CTR model $\mathcal{M}_{sp}$: $\hat{y}_{sp} = \mathcal{M}_{sp}([e_i]_{i=1}^n)$. Then, these prediction scores are aggregated with the original shared prediction scores:

$$\hat{y}_a = \hat{y}_{sh} + \hat{y}_{sp} \tag{10}$$

Then, we use $\hat{y}_a$ to compute the CTR training loss and optimize the newly added parameters $\mathcal{E}_{sp}$ and $\mathcal{M}_{sp}$ to capture specific information. Note that during the optimization in the second stage, the shared parameters $\mathcal{E}_{sh}$ and $\mathcal{M}_{sh}$ are frozen to allow the model to better focus on learning specific knowledge and to keep the shared knowledge from being disrupted.

4.4 Adaptive Distribution Re-balancing and Calibration

4.4.1 Adaptive Distribution Rebalancing. It is worth noting that the distribution alignment module is performed at the representation level. However, at the sample quantity level, there may be an inherent imbalance between long-tail users and head users. Ignoring this imbalance could lead the model to overly focus on the distribution of head users, resulting in poor performance for long-tail users.

Therefore, we use a sample reweighting method [19] to balance the distributional base difference between long-tail users and head users. Specifically, we assign a weight w_i for the CTR training loss of each group, and calculate the total loss as

$$\mathcal{L}_{ctr\_rebalancing} = \sum_{i=0}^{1} \frac{w_i}{N_i} \sum_{j=0}^{N_i - 1} (y \log \hat{y} + (1 - y) \log(1 - \hat{y})) \tag{11}$$

where N_i is the sample number of the corresponding group, and w_i is updated at each batch by:

$$w_i \leftarrow \frac{w_i \cdot \exp(\eta \cdot \mathcal{L}_i(\theta))}{\sum_j w_j \cdot \exp(\eta \cdot \mathcal{L}_j(\theta))} \tag{12}$$

where η is a hyperparameter, and $\mathcal{L}_i(\theta)$ is the mean loss of the corresponding group.

4.4.2 Adaptive Distribution Calibration. Although we use a distribution balancing loss to balance the weights of long-tail and head users, ensuring the model learns equitably for both groups, this method may result in a mismatch between the training data distribution and the testing data distribution. Unlike classification tasks that focus on accuracy, CTR tasks requiring precise and unbiased probability estimation can be significantly affected by subtle inconsistencies in distribution, potentially leading to systemic bias in the final probability estimates.

Therefore, after completing the alignment and residual learning, we use a probability calibration module to prevent potential biases. Specifically, considering that long-tail and head users are treated differently, we also introduce corresponding adaptive parameters during distribution calibration.

Specifically, we use Beta Calibration [13] for calibration, which requires an additional dataset separate from the training set. In practice, we use the validation set for calibration.

$$\hat{y}_c = \frac{1}{1 + \frac{(1-\hat{y}_a)^{b+b_g}}{e^{c+c_g}\,\hat{y}_a^{a+a_g}}} \tag{13}$$

The corresponding parameters are learning via a binary cross entropy loss:

$$\mathcal{L}_{ctr\_calib} = \sum_i y \log \hat{y}_c + (1-y)\log(1-\hat{y}_c) \tag{14}$$

5 Experiment

To validate the effectiveness of our proposed method, we conduct a series of experiments to answer the following research questions:

- **RQ1:** How does our method effectively improve the experience of long-tail users in the CTR prediction task compared to state-of-the-art methods?
- **RQ2:** How does each component of our method affect the accuracy and fairness?
- **RQ3:** How do the hyperparameters affect the performance of long-tail users?
- **RQ4:** Does the hierarchical distribution alignment technique effectively align the distributions of long-tail and head users?
- **RQ5:** Does our method show effectiveness in the online system?

Additional experimental analyses, such as the performance analysis across user groups, are provided in the appendix.

5.1 Datasets and Settings

5.1.1 Datasets. We conduct experiments on two commonly used datasets: **Movielens**[2] and **Taobao**[3]. Note that since we need to perform user analysis, some of the more commonly used CTR datasets (such as Criteo and Avazu) cannot be used for the experiments.

Movielens. This dataset contains 1 million movie ratings with user and item profiles. Following previous work, the ratings above 4 are treated as positive feedback, and the others are treated as negatives. We use 5-core filtering, which is a common setting.

Taobao. This dataset contains 26 million ad display/click logs from the website of Taobao for 8 days, with user and ad profiles.

For all datasets, we remove irrelevant users and items and then randomly divide all interactions into training, validation, and test sets in the ratio of 7:1:2. Statistics of datasets are shown in Table 1.

Table 1: Statistics of the processed datasets.

Dataset	#Users	#Items	#Exposure	CTR
Movielens	6,025	3,126	993,801	0.577
Taobao	61,870	49,476	4,951,767	0.085

5.1.2 Metrics. For accuracy metrics, we adopt two metrics that are widely used in CTR prediction tasks. The first metric is area under ROC curve (**AUC**) that measures the pairwise ranking performance of the CTR prediction model. The second one is the log loss (**Logloss**), which measures the accuracy of the absolute value of the CTR prediction.

As mentioned in the preliminary section, for long-tail users, we evaluate their performance from the perspectives of both accuracy and fairness. For accuracy, we use the average log loss of long-tail users (**GP-LT**) as the metric, and for fairness, we measure it using the difference in average log loss between long-tail users and head users (**GAP**).

5.1.3 Baselines. The following state-of-the-art methods are our baselines for improving long-tail user performance in CTR prediction:

- **Vanilla**: the backbone model without any change.
- **Calibration**[13]: the backbone model with Beta Calibration.
- **IPW**[19]: a static reweighting method using propensity scores.
- **SDRO**[27]: a dynamic reweighting method using streaming distributionally robust optimization.
- **MLoRA**[32]: a framework based on multiple LoRAs for multi-domain recommendation. Here, we treat different user groups as different domains.
- **Ours**: our method using hierarchical distribution alignment and residual learning with adaptive distribution calibration.

The above methods can be applied to different CTR models. We consider the following popular CTR models that focus on feature interactions as backbones:

- **MLP**: the classical fully connected neural networks.

[2]https://grouplens.org/datasets/movielens/1m/
[3]https://www.kaggle.com/datasets/pavansanagapati/ad-displayclick-data-on-taobaocom

Dataset		MovieLens				Taobao			
Backbone	**Method**	**AUC↑**	**Logloss↓**	**GP-LT↓**	**GAP↓**	**AUC↑**	**Logloss↓**	**GP-LT↓**	**GAP↓**
MLP	**Vanilla**	0.8055	0.5274	0.5777	0.0530	0.6425	0.2800	0.6221	0.3178
	Calibration	0.8055	0.5246	0.5687	0.0456	0.6425	0.2800	0.6209	0.3163
	IPW	0.8054	0.5272	0.5768	0.0526	0.6429	0.2799	0.6206	0.3164
	SDRO	0.8060	0.5264	0.5743	0.0498	0.6416	0.2802	0.6171	0.3146
	MLoRA	0.8089	0.5235	0.5754	0.0554	0.6412	0.2847	0.6197	0.3054
	Ours	**0.8102***	**0.5191****	**0.5599**	**0.0420***	**0.6571****	**0.2778****	**0.6004****	**0.2928****
WideDeep	**Vanilla**	0.8099	0.5222	0.5718	0.0527	0.6499	0.2790	0.6116	0.3078
	Calibration	0.8099	0.5194	0.5634	0.0460	0.6499	0.2789	0.6152	0.3108
	IPW	0.8087	0.5237	0.5734	0.0525	0.6497	0.2790	0.6111	0.3076
	SDRO	0.8085	0.5228	0.5684	0.0470	0.6482	0.2795	0.6088	0.3075
	MLoRA	0.8096	0.5221	0.5713	0.0518	0.6479	0.2829	0.6137	0.3015
	Ours	**0.8109**	**0.5182***	**0.5610****	**0.0440****	**0.6566****	**0.2778****	**0.5993****	**0.2923****
DeepFM	**Vanilla**	0.8118	0.5212	0.5692	0.0489	0.6484	0.2792	0.6143	0.3102
	Calibration	0.8118	0.5171	0.5584	**0.0411**	0.6484	0.2791	0.6168	0.3124
	IPW	0.8118	0.5215	0.5702	0.0499	0.6481	0.2793	0.6136	0.3099
	SDRO	0.8123	0.5200	0.5656	0.0460	0.6484	0.2793	0.6112	0.3088
	MLoRA	0.8122	0.5207	0.5691	0.0496	0.6471	0.2830	0.6147	0.3021
	Ours	**0.8143****	**0.5143****	**0.5555****	0.0416	**0.6542****	**0.2783****	**0.6000****	**0.2930****
DCNV2	**Vanilla**	0.8116	0.5248	0.5691	0.0447	0.6520	0.2792	0.6156	0.3095
	Calibration	0.8116	0.5225	0.5632	**0.0407**	0.6520	0.2786	0.6119	0.3068
	IPW	0.8111	0.5257	0.5700	0.0448	0.6517	0.2792	0.6158	0.3100
	SDRO	0.8112	0.5252	0.5687	0.0439	0.6505	0.2792	0.6163	0.3110
	MLoRA	0.8127	0.5246	0.5774	0.0559	0.6487	0.2829	0.6118	0.2986
	Ours	**0.8135**	**0.5152****	**0.5578****	0.0437**	**0.6571****	**0.2778****	**0.6005****	**0.2927****

Table 2: Performance comparison of different methods on MovieLens and Taobao datasets. Bold for the best and underline for the second best. */** indicate $p \leq 0.05/0.01$ for the t-test of our method vs. the best baseline. ↑/↓ means the higher/lower the better. AUC and Logloss for the overall accuracy. GP-LT for the accuracy of tail user. GAP for the fairness.

- **WideDeep**[2]: the model combining the linear regression and deep neural networks.
- **DeepFM**[7]: the model combining the factorization machines and deep neural networks.
- **DCNV2**[23]: the model using deep and cross networks to learn effective explicit and implicit feature crosses.

5.1.4 Implement Details. The embedding size is 64 for all the methods. Since our method involves two CTR models, to ensure fairness in model capacity, we use half the embedding size for each model. The MLP model in all the backbones is a simple three-layer model with hidden units [64, 32] and ReLU activation. Adam[12] is used as the optimizer. The batch size is set to 1024 for all the methods. The hyperparameters for all the methods are tuned based on validation datasets with an early stopping strategy. We repeat each experiment 5 times with different random seeds and report the average results and make the statistical tests.

5.2 RQ1: Overall Performance

As shown in Table 2, our proposed method significantly enhances the accuracy and fairness of long-tail users compared to all the baselines, while also maintaining a comparable or even better overall accuracy. There are some further observations.

In terms of long-tail user performance (**GP-LT**), we observe that: (i) Although the calibration method is not specifically designed for the long-tail user problem, it significantly improves the performance for long-tail users on the Movielens dataset. This suggests that the original model's predictions for long-tail users may have more severe bias; (ii) Two reweighting methods, IPW and SDRO, indeed effectively improve the tail users, with SDRO achieving better performance due to its consideration of the optimization dynamics; (iii) MLoRA does not consistently improve long-tail user performance, which may be because it does not explicitly learn shared knowledge or adjust sample weights. (iv) Our method effectively improves the performance of long-tail users, owing to

our hierarchical learning of shared and unique knowledge, as well as the re-balance of sample distribution through rebalancing and calibration.

In terms of fairness (**GAP**), we observe that all the vanilla models exhibit significant performance differences between long-tail users and head users on both datasets. Especially on the Taobao dataset, the logloss for long-tail users is nearly twice that of head users, reflecting a significant fairness issue. Besides, fairness is somewhat consistent with the improvement of long-tail user performance, meaning that the enhancement of long-tail users' performance can also help promote the improvement of overall fairness.

In terms of overall accuracy (**AUC** and **Logloss**), we observe that two reweighting methods, IPW and SDRO, may lead to a significant loss in overall accuracy. We argue that optimizing the long-tail problem solely through sample weights does not address the insufficient knowledge of long-tail users, leading to suboptimal performance. In contrast, our method consistently improves the overall accuracy, owing to our hierarchical learning of shared and unique knowledge, as well as the group-wise adaptive calibration.

To conclude, these results validate that our method can effectively enhances the accuracy and fairness of long-tail users while maintaining similar or even better overall accuracy.

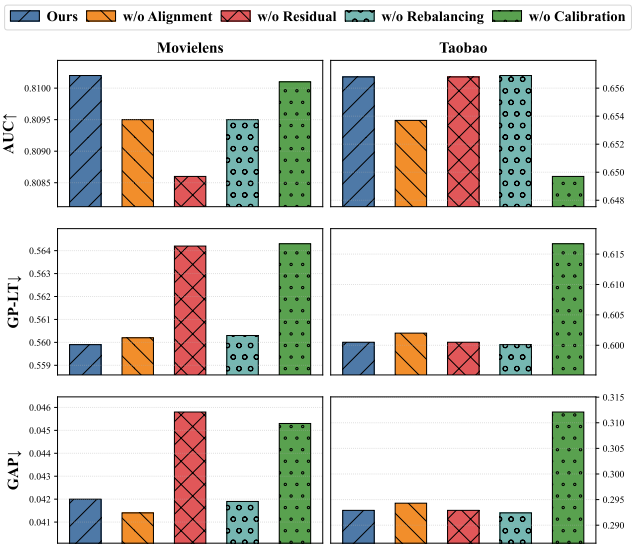


Figure 3: Ablation for our method with MLP as backbone. All the metrics are the lower the better except for AUC.

5.3 RQ2: Ablation Study

We conduct ablation studies to assess the effectiveness of the four components within our method. Specifically, four variants are examined: removing hierarchical distribution alignment (**w/o Alignment**), removing hierarchical residual learning (**w/o Residual**), removing distribution re-balancing loss (**w/o Rebalancing**), and removing adaptive distribution calibration (**w/o Calibration**).

As shown in Figure 3, each module demonstrates efficacy in enhancing the performance of long-tail users. The effectiveness of these modules exhibits some variation contingent upon dataset characteristics. We find that alignment and calibration both demonstrate effective results on the two datasets, with calibration demonstrating more significant improvements for long-tail users. We believe this reflects the model's greater bias toward long-tail users during training. Unlike the alignment and calibration modules, the residual and re-weighting modules had a smaller impact on the Taobao dataset. This is because, in practice, we find that the Taobao dataset exhibits the "one epoch" phenomenon [37], which causes the model to easily overfit during the residual stage and makes it difficult to learn stable sample weights. How to further address this issue is left for future work.

5.4 RQ3: Hyperparameter Analysis

We further analyze the influence of the involved hyperparameters on the performance of long-tail users, specifically λ_1 in probability feature mapping, λ_2 in adversarial learning, and η in adaptive group rebalancing. The results are present in Table 4. For all the parameters, the performance of long-tail users tends to initially improve and subsequently decline as the parameter value increases. A similar pattern is observed in the performance gap between long-tail and head users. For λ_1, λ_2, if they are too small, the model may fail to effectively learn shared knowledge, leading to worse performance for long-tail users. On the other hand, if they are too large, it may result in the CTR loss being less dominant, making it difficult to effectively retain task-related information, which also causes poor performance for long-tail users. For η, which controls the emphasis on long-tail user samples during model training, if it is too small, the lower weight given to long-tail user samples will lead to poor performance for long-tail users. If it is too large, it may cause the model to ignore shared knowledge between head users and long-tail users, which also leads to poor performance for long-tail users.

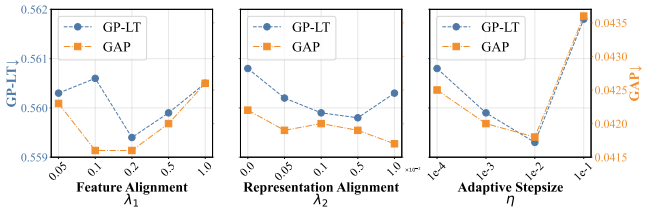


Figure 4: Hyperparameter analysis on the Movielens dataset with MLP as backbone. Both metrics are the lower the better.

5.5 RQ4: Distribution Alignment Analysis

We conducted experiments to verify whether the hierarchical distribution alignment module effectively achieves the goal of distribution alignment. We use an additional discriminator to determine whether a sample belongs to a head user or a long-tail user based on the sample's representation E_{sh}. The discriminator's prediction performance, measured by AUC, is used as the criterion for whether the distribution is aligned, which is a common evaluation metric [29]. The smaller the difference between the two distributions, the less the discriminator can distinguish the distribution information,

and the closer the AUC will be to 0.5. Specifically, we use DeepFM as the discriminator.

As shown in Figure 5, we can observe that hierarchical distribution alignment effectively aligns the distributions of long-tail users and head users. We also have the following additional observations: there are differences in the representation distributions of tail and head users after training with the CTR task, which indicates that tail and head users indeed possess differentiated knowledge. Besides, both probabilistic feature mapping and adversarial learning can align the distributions, and combining both achieves the best alignment, thus verifying the effectiveness of hierarchical distribution alignment.

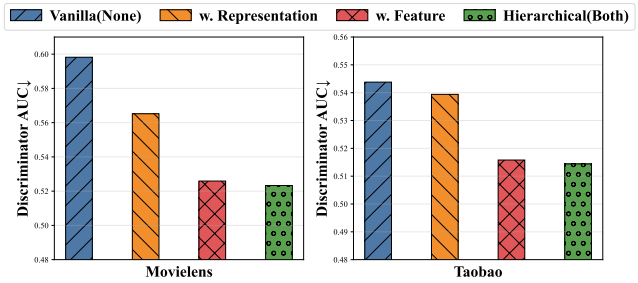


Figure 5: Alignment analysis for our method with MLP as backbone. The AUC of the discriminator is the lower the better.

5.6 RQ5: Online A/B Experiment

To examine the effectiveness of our method in the online setting, we conduct online A/B testing on a short video recommendation platform. The online A/B experiment lasted for 6 days with 10% of the user traffic. Table 3 shows the experimental results. We define Finish-View-Through Rate (FVTR), which equals 1 if the watching time reaches the 75th percentile watching time and 0 otherwise. It can be observed that our method effectively improves the performance for long-tail users and enhances the overall performance.

Online Metric	App Usage Time	FVTR
All users	+0.201%	+0.617%
Head users	+0.128%	+0.538%
Tail users	+0.851%	+1.329%

Table 3: Results of the online A/B testing experiment. All the performance gains are statistically significant at $p < 0.05$.

6 Conclusion

In this paper, we attempt to improve the performance of long-tail users in the CTR prediction task from the perspective of distribution alignment. By tackling two key challenges—limited informative data and imbalanced sample distributions—our proposed framework significantly enhances the performance of long-tail user groups. Leveraging hierarchical distribution alignment and residual learning, our method extracts shared knowledge between head and long-tail users while preserving group-specific knowledge. Furthermore, dynamic reweighting and the introduction of a distribution calibration module effectively address the data imbalance and mitigate bias in model training. Experimental results on public datasets and an online experiment validate the superiority of our approach in improving long-tail user performance while maintaining similar or even better overall performance.

References

[1] Lei Chen, Le Wu, Kun Zhang, Richang Hong, Defu Lian, Zhiqiang Zhang, Jun Zhou, and Meng Wang. 2023. Improving Recommendation Fairness via Data Augmentation. In *Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23)*. Association for Computing Machinery, New York, NY, USA, 1012–1020. doi:10.1145/3543507.3583341

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) *(DLRS 2016)*. Association for Computing Machinery, New York, NY, USA, 7–10. doi:10.1145/2988450.2988454

[3] Virginie Do, Sam Corbett-Davies, Jamal Atif, and Nicolas Usunier. 2022. Online certification of preference-based fairness for personalized recommender systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 6532–6540.

[4] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 69–78.

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research* 17, 59 (2016), 1–35. http://jmlr.org/papers/v17/15-239.html

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (Oct. 2020), 139–144. doi:10.1145/3422622

[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 1725–1731.

[8] Zhongxuan Han, Chaochao Chen, Xiaolin Zheng, Meng Li, Weiming Liu, Binhui Yao, Yuyuan Li, and Jianwei Yin. 2024. Intra- and Inter-group Optimal Transport for User-Oriented Fairness in Recommender Systems. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 8 (Mar. 2024), 8463–8471. doi:10.1609/aaai.v38i8.28689

[9] Zhongxuan Han, Chaochao Chen, Xiaolin Zheng, Li Zhang, and Yuyuan Li. 2024. Hypergraph Convolutional Network for User-Oriented Fairness in Recommender Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 903–913. doi:10.1145/3626772.3657737

[10] Rashidul Islam, Kamrun Naher Keya, Ziqian Zeng, Shimei Pan, and James Foulds. 2021. Debiasing career recommendations with neural fair collaborative filtering. In *Proceedings of the Web Conference 2021*. 3779–3790.

[11] Mesut Kaya, Derek Bridge, and Nava Tintarev. 2020. Ensuring fairness in group recommendations by rank-sensitive balancing of relevance. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 101–110.

[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[13] Meelis Kull, Telmo Silva Filho, and Peter Flach. 2017. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 623–631. https://proceedings.mlr.press/v54/kull17a.html

[14] Jiayu Li, Hanyu Li, Zhiyu He, Weizhi Ma, Peijie Sun, Min Zhang, and Shaoping Ma. 2024. ReChorus2.0: A Modular and Task-Flexible Recommendation Library. In *Proceedings of the 18th ACM Conference on Recommender Systems* (Bari, Italy) *(RecSys '24)*. Association for Computing Machinery, New York, NY, USA, 454–464. doi:10.1145/3640457.3688076

[15] Roger Zhe Li, Julián Urbano, and Alan Hanjalic. 2021. Leave No User Behind: Towards Improving the Utility of Recommender Systems for Non-Mainstream Users. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (Virtual Event, Israel) *(WSDM '21)*. Association for Computing Machinery, New York, NY, USA, 103–111. doi:10.1145/3437963.3441769

[16] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2021. User-oriented fairness in recommendation. In *Proceedings of the Web Conference 2021*. 624–632.

[17] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2021. Towards Personalized Fairness Based on Causal Notion. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1054–1063. doi:10.1145/3404835.3462966

[18] Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. 2024. LLM-ESR: Large Language Models Enhancement for Long-tailed Sequential Recommendation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[19] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2020. Distributionally Robust Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=ryxGuJrFvS

[20] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. 2019. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment* 2019, 12 (2019), 124020.

[21] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. 2018. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[22] Liangcai Su, Junwei Pan, Ximei Wang, Xi Xiao, Shijie Quan, Xihua Chen, and Jie Jiang. 2024. STEM: unleashing the power of embeddings for multi-task recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 9002–9010.

[23] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 1785–1797. doi:10.1145/3442381.3450078

[24] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. 2023. A Survey on the Fairness of Recommender Systems. *ACM Trans. Inf. Syst.* 41, 3, Article 52 (Feb. 2023), 43 pages. doi:10.1145/3547333

[25] Yifan Wang, Peijie Sun, Weizhi Ma, Min Zhang, Yuan Zhang, Peng Jiang, and Shaoping Ma. 2024. Intersectional Two-sided Fairness in Recommendation. In *Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 3609–3620. doi:10.1145/3589334.3645518

[26] Tianxin Wei, Fuli Feng, Jiawei Chen, Ziwei Wu, Jinfeng Yi, and Xiangnan He. 2021. Model-Agnostic Counterfactual Reasoning for Eliminating Popularity Bias in Recommender System. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1791–1800. doi:10.1145/3447548.3467289

[27] Hongyi Wen, Xinyang Yi, Tiansheng Yao, Jiaxi Tang, Lichan Hong, and Ed H. Chi. 2022. Distributionally-Robust Recommendations for Improving Worst-Case User Experience. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 3606–3610. doi:10.1145/3485447.3512255

[28] Junda Wu, Cheng-Chun Chang, Tong Yu, Zhankui He, Jianing Wang, Yupeng Hou, and Julian McAuley. 2024. CoRAL: Collaborative Retrieval-Augmented Large Language Models Improve Long-tail Recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 3391–3401. doi:10.1145/3637528.3671901

[29] Le Wu, Lei Chen, Pengyang Shao, Richang Hong, Xiting Wang, and Meng Wang. 2021. Learning fair representations for recommendation: A graph-based perspective. In *Proceedings of the Web Conference 2021*. 2198–2208.

[30] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. 2017. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2272–2281.

[31] Hao Yang, Zhining Liu, Zeyu Zhang, Chenyi Zhuang, and Xu Chen. 2023. Towards Robust Fairness-Aware Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys '23)*. Association for Computing Machinery, New York, NY, USA, 211–222. doi:10.1145/3604915.3608784

[32] Zhiming Yang, Haining Gao, Dehong Gao, Luwei Yang, Libin Yang, Xiaoyan Cai, Wei Ning, and Guannan Zhang. 2024. MLoRA: Multi-Domain Low-Rank Adaptive Network for CTR Prediction. In *Proceedings of the 18th ACM Conference on Recommender Systems* (Bari, Italy) *(RecSys '24)*. Association for Computing Machinery, New York, NY, USA, 287–297. doi:10.1145/3640457.3688134

[33] Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. *Advances in neural information processing systems* 30 (2017).

[34] Xiaoyu Zhang, Shaoyun Shi, Yishan Li, Weizhi Ma, Peijie Sun, and Min Zhang. 2024. Feature-Enhanced Neural Collaborative Reasoning for Explainable Recommendation. *ACM Trans. Inf. Syst.* 43, 1, Article 7 (Nov. 2024), 33 pages. doi:10.1145/3690381

[35] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H. Chi. 2021. A Model of Two Tales: Dual Transfer Learning Framework for Improved Long-tail Item Recommendation. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 2220–2231. doi:10.1145/3442381.3450086

[36] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 11–20. doi:10.1145/3404835.3462875

[37] Zhao-Yu Zhang, Xiang-Rong Sheng, Yujing Zhang, Biye Jiang, Shuguang Han, Hongbo Deng, and Bo Zheng. 2022. Towards Understanding the Overfitting Phenomenon of Deep Click-Through Rate Models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 2671–2680. doi:10.1145/3511808.3557479

A Proofs of Theorems

A.1 Proof of Theorem 4.1

Theorem A.1 (Discrete Optimization Limit). *Let $\mathcal{P}$ be any fixed data distribution and $G \in \{0,1\}$ be a binary group attribute. If there exists a categorical feature X such that $X \not\perp G$, then there exists a constant $c > 0$, such that for any non-trivial embedding table parameters (i.e., not all embeddings are the same), the mutual information $\mathcal{I}(E_{sh}, G)$ for E_{sh} in the form of Eq.(8) has $\mathcal{I}(E_{sh}, G) > c$. This implies that it is impossible to learn a perfect shared representation.*

Proof. This is mainly because the embedding table inherently eliminates group-specific information by merging different feature values, and within the limited discrete solution space, an optimal invariant solution may not exist.

We begin by reducing the problem to a simpler one: consider only the embedding of a single categorical feature X, and its mutual information with the group attribute G. If $\mathcal{I}(E_X, G) > 0$, then adding more features (e.g., concatenation with others) cannot eliminate this dependency, and the full shared representation E_{sh} must also satisfy $\mathcal{I}(E_{\text{sh}}, G) > 0$.

Assume $G \in \{0,1\}$, and X takes n discrete values. Then the joint distribution $P_{X,G}$ can be represented as an $n \times 2$ matrix M, where $M_{ij} = \mathbb{P}(X = i, G = j)$.

We note that mapping the feature X to an m-dimensional embedding through the embedding table essentially amounts to mapping a one-dimensional discrete variable X to an m-dimensional discrete variable $E = (e_1, e_2, ..., e_m)$, as the embedding table is finite and the resulting embedding has only a finite number of possible values. The mutual information between the m-dimensional discrete variable E and the one-dimensional discrete variable G is:

$$I(E;G) = \sum_{e \in E} \sum_{g \in G} p(e,g) \log \left(\frac{p(e,g)}{p(e)p(g)} \right) \tag{15}$$

We can easily observe that the way the embedding table affects mutual information $I(E;G)$ depends on how it maps X. If the mapping from X to E is a constant mapping, meaning all values of X

are mapped to the same embedding, then although $I(E, G) = 0$, but the embedding table is trivial and does not contain any meaningful information.

The influence of embedding table parameters on mutual information is equivalent to performing row partitioning over the joint distribution matrix. Since $X \not\perp G$, the matrix M has **rank 2**. This implies that for any **non-trivial partition** of the n rows into $k > 1$ groups, if we sum the rows within each group to get k group-level vectors, **at least two of them are not collinear** (i.e., linearly independent).

Therefore, if the embedding table is non-trivial (i.e., it induces a partition with $k > 1$), the resulting embedded representations must preserve some information about group G, meaning $\mathcal{I}(E_X, G) > 0$. This lower bound must hold for some constant $c > 0$, depending on $\mathcal{P}$, which completes the proof.

□

A.2 Proof of Theorem 4.2

Theorem A.2 (improvement property). *Adding the probabilistic feature mapping into optimization through Eq.(3) before the adversarial learning in Eq.(5) does not lead to an increase in the minimum value of $\mathcal{I}(E_{sh}, G)$, and there exists a case where the minimum value is reduced to 0 after adding probabilistic feature mapping.*

Proof. In Eq.(3), when we set λ_1 to 0, the probability transition matrix will be the identity matrix. In this case, the minimum value $\mathcal{I}(E_{sh}, G)$ is just the minimum value without probabilistic feature mapping. Therefore, adding a probability transition matrix in the optimization does not lead to an increase in the minimum value of $\mathcal{I}(E_{sh}, G)$.

Additionally, we can construct an example where adding the probability transition module can indeed lower the minimum value. For simplicity, let's assume that the dataset has only one feature $X \in \{A, B, C\}$, and the group identifier $G \in \{0, 1\}$. The probability distribution (X, G) on the dataset is shown in the table below:

Probability Distribution	G=0	G=1
X=A	$\frac{3}{30}$	$\frac{7}{30}$
X=B	$\frac{2}{30}$	$\frac{8}{30}$
X=C	$\frac{6}{30}$	$\frac{4}{30}$

The mapping function space is finite. For this example, we can easily calculate that the mapping with the smallest mutual information is to mapping B and C to the same embedding, and the corresponding mutual information value is 0.0049.

Thus, for this example, for any non-trivial embedding table parameters, we have $\mathcal{I}(E_{sh}, G) \geq 0.0049$.

Let's consider the following transition matrix.

Then the probability distribution matrix after the transition will be:

For any embedding table parameters, we have:

$$I(E_{sh}, G) = I(X_t, G) = 0$$

.

□

Transition Distribution	$X_t = M$	$X_t = N$
X=A	0	1
X=B	$\frac{7}{12}$	$\frac{5}{12}$
X=C	$\frac{5}{12}$	$\frac{7}{12}$

Probability Distribution	G=0	G=1
$X_t = M$	$\frac{11}{90}$	$\frac{19}{90}$
$X_t = N$	$\frac{22}{90}$	$\frac{38}{90}$

B Additional Experiments

B.1 Comparison with Multi-task Baselines

Multi-task methods often involve both shared parameters and task-specific parameters, which bears some similarity to our approach. We compared our method with the current state-of-the-art multi-task method, STEM [22]. Considering that most existing multi-task learning studies only use MLP as the backbone, we also conducted our comparison using MLP only. As shown in the table, although multi-task learning provides a slight improvement for long-tail users, the gain is not significant. We attribute this to the lack of explicit guidance in learning shared versus task-specific knowledge, as well as the absence of consideration for the varying importance of different user groups during optimization.

Table 4: Performance comparison with MLP as the backbone on the Movielens dataset.

Method	AUC	Logloss	GP-LT	GAP
Vanilla	0.8055	0.5274	0.5777	0.0530
STEM	0.8050	0.5265	0.5760	0.0527
Ours	**0.8102**	**0.5191**	**0.5599**	**0.0420**

B.2 Ablation on Frozen Parameters

During the residual learning stage, we freeze shared knowledge parameters in order to allow the model to better focus on learning specific knowledge and to keep the shared knowledge from being disrupted. The below ablation experimental results show that freezing the parameters indeed leads to better performance.

Table 5: Ablation on frozen parameters with MLP as the backbone on the Movielens dataset.

	AUC↑	Logloss↓	GP-LT↓	GAP↓
Vanilla	0.8055	0.5274	0.5777	0.0530
Ours w/o frozen	0.8073	0.5217	0.5649	0.0449
Ours	**0.8102**	**0.5191**	**0.5599**	**0.0420**

B.3 User Group Analysis

We divided users into five groups based on their activity levels and conducted further performance analysis across these user groups. As shown in the table below, our method effectively improves the experience of users in the long tail, while not significantly degrading the performance for head users.

Table 6: Logloss across different user activity groups of MLP on the Taobao dataset. G1 is the least active user group, while G5 is the most active.

	G1	G2	G3	G4	G5
Vanilla	0.8014	0.7202	0.6264	0.4921	**0.3052**
Ours	**0.7586**	**0.6865**	**0.6009**	**0.4759**	0.3069
Impr.	-5.3%	-4.6%	-4.0%	-3.2%	+0.05%