From Entity Reliability to Clean Feedback: An Entity-Aware Denoising Framework Beyond Interaction-Level Signals

Ze Liu
University of Science and Technology of China
Hefei, China
liuze2120@mail.ustc.edu.cn

Xianquan Wang
University of Science and Technology of China
Hefei, China
wxqcn@mail.ustc.edu.cn

Shuochen Liu
University of Science and Technology of China
Hefei, China
shuochenliu@mail.ustc.edu.cn

Jie Ma
University of Science and Technology of China
Hefei, China
master@mail.ustc.edu.cn

Huibo Xu
University of Science and Technology of China
Hefei, China
xhbxhb@mail.ustc.edu.cn

Yupeng Han
University of Science and Technology of China
Hefei, China
yupenghan@mail.ustc.edu.cn

Kai Zhang*
University of Science and Technology of China
Hefei, China
kkzhang08@ustc.edu.cn

Jun Zhou
Zhejiang University
Hangzhou, China
junzhougucas@gmail.com

Abstract

Implicit feedback is central to modern recommender systems but is inherently noisy, often impairing model training and degrading user experience. At scale, such noise can mislead learning processes, reducing both recommendation accuracy and platform value. Existing denoising strategies typically overlook the entity-specific nature of noise while introducing high computational costs and complex hyperparameter tuning. To address these challenges, we propose **EARD** (**E**ntity-**A**ware **R**eliability-**D**riven Denoising), a lightweight framework that shifts the focus from interaction-level signals to entity-level reliability. Motivated by the empirical observation that training loss correlates with noise, EARD quantifies user and item reliability via their average training losses as a proxy for reputation, and integrates these entity-level factors with interaction-level confidence. The framework is **model-agnostic**, **computationally efficient**, and requires **only two intuitive hyperparameters**. Extensive experiments across multiple datasets and backbone models demonstrate that EARD yields substantial improvements over state-of-the-art baselines (e.g., up to 27.01% gain in NDCG@50), while incurring negligible additional computational cost. Comprehensive ablation studies and mechanism analyses further confirm EARD's robustness to hyperparameter choices and its practical scalability. These results highlight the importance of entity-aware reliability modeling for denoising implicit feedback and pave the way for more robust recommendation research.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Learning from implicit feedback.*

Keywords

Recommender systems; Implicit feedback; Denoising methods

*Corresponding author.

WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792128

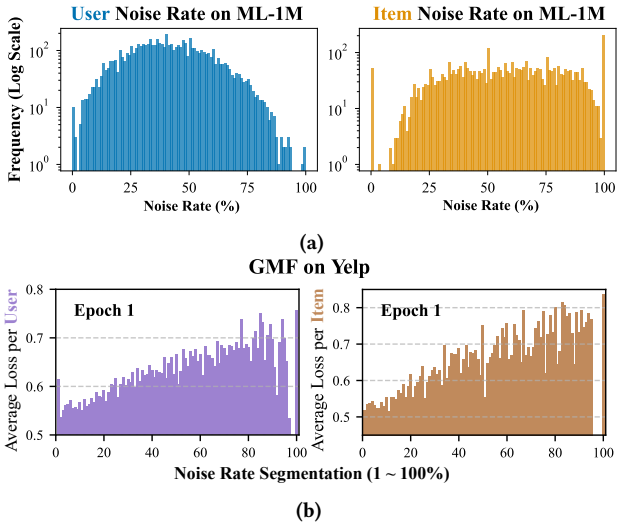

Figure 1: (a) User (blue) and item (yellow) noise rate distributions on ML-1M. Following ADT [21], ratings ≤ 3 are labeled as noise. The wide spread indicates substantial variation in entity reliability. (b) Average loss of user (purple) and item (brown) groups with different noise levels on Yelp using GMF. Average loss increases with noise rate.

ACM Reference Format:
Ze Liu, Xianquan Wang, Shuochen Liu, Jie Ma, Huibo Xu, Yupeng Han, Kai Zhang, and Jun Zhou. 2026. From Entity Reliability to Clean Feedback: An Entity-Aware Denoising Framework Beyond Interaction-Level Signals. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792128

1 Introduction

Recommender systems are essential to modern information platforms, supporting personalized content delivery in e-commerce and social media. These systems increasingly rely on large-scale implicit feedback, such as clicks and viewing histories, to model user preferences. Implicit signals are abundant and require no explicit input, making them well-suited for industrial-scale deployment.

However, implicit feedback is inherently noisy, containing both *false positives* (e.g., accidental clicks or clickbait-induced interactions) and *false negatives* (non-interactions that do not imply disinterest, often due to exposure bias). Such noise can misguide model training and degrade recommendation accuracy. To mitigate this, numerous denoising methods have been proposed to address the noise inherent in implicit feedback for recommendation systems. Prevailing strategies typically involve re-weighting training instances based on their loss values [21], pruning noisy edges from the user-item interaction graph [19], or leveraging more complex mechanisms like cross-model disagreement [6, 24]. Recommendation systems rely on massive amounts of data and incur substantial training costs, so any additional denoising mechanism must be lightweight, efficient, and compatible with the main training pipeline without interfering with or slowing down the original process. This raises a fundamental and critical research question: How can we design a denoising framework that is not only effective at identifying noise sources but also practical for large-scale, real-world deployment? Answering this question requires overcoming three major, interrelated challenges that plague existing methods:

Lack of Entity-Aware Modeling. Most existing methods address noise within the vast and sparse user–item interaction space, yet they typically analyze it only at the interaction level. A critical oversight in these approaches is the failure to decouple the origins of noise by attributing it distinctly to the user or the item. For instance, noise can stem from the user side, such as from users who are merely 'wandering' and whose clicks do not reflect genuine preference. Conversely, it can originate from the item side, where certain items act as 'clickbait,' engineered to attract interactions that are not indicative of true interest. The necessity of this decoupled perspective is substantiated by our analysis of user and item noise rates across the ML-1M, Yelp, and AmazonBook datasets. As shown in Figure 1a, noise rates for both entities span the full 0–100% range with substantial variability, a pattern confirmed in other datasets (Appendix Figure 6). These findings refute the assumption of uniform entity reliability and highlight the need for entity-aware denoising that independently models user and item reliability to better assess and suppress noisy interactions.

High Computational Overhead. To improve denoising effectiveness, several methods, including SGDL [6], DeCA [24], and BOD [25], adopt complex optimization strategies such as meta-learning frameworks, second-order gradient computations, and multi-path auxiliary networks. While these approaches can achieve strong performance in small- or medium-scale experiments, they introduce substantial training and inference costs. This overhead becomes prohibitive in industrial-scale scenarios with tens of millions of interactions, limiting their practicality for real-time recommendation systems that demand both efficiency and scalability.

Excessive Hyperparameter Complexity. Many existing methods require multiple hyperparameters to manage loss weighting, sample selection, or optimization. For example, DeCA, UDT, and SGDL introduce three to five core hyperparameters. This raises tuning costs, undermines cross-dataset generalization, and extensive hyperparameter search is often impractical in real-world settings. Prior work [4] shows that simpler models can match more complex ones under fair evaluation, motivating denoising methods that minimize manual tuning and remain robust under default settings.

To address these challenges, we estimate an entity's noise level using a simple yet effective proxy. Specifically, for each user (or item), we compute its average training loss over all associated interactions in an epoch. This approach is inspired by prior works [1, 8, 13, 21], which show that noisy samples tend to incur higher losses. To validate this assumption, we grouped users and items by noise rate bins and computed their average losses during the first epoch of GMF on the Yelp dataset. As shown in Figure 1b, the average loss of entities generally increases with their noise rate. Additional evidence in Appendix D (Figures 7a and 7b) confirms this trend.

Based on this observation, we propose a lightweight three-stage pipeline to estimate fine-grained confidence scores: (1) compute the average loss for each entity with minimal overhead; (2) weight entity-level scores using interaction-level weights derived from the empirical cumulative distribution function (ECDF) of individual losses; (3) transform losses into reliability scores via a linear mapping parameterized by α and β. This design captures both entity reliability and interaction uncertainty, while remaining model-agnostic, efficient, and requiring minimal hyperparameter tuning.

Our key contributions are as follows:

1) Entity-Aware denoising. We propose EARD, a novel and effective framework that explicitly incorporates both user and item reliability to better detect noise in implicit feedback.

2) Scalable and efficient design. Our method computes entity- and interaction-level weights with negligible overhead, enabling application to large-scale industrial datasets.

3) Minimal hyperparameter requirement. The framework introduces only two intuitive hyperparameters, α and β, with a smooth and unimodal performance landscape that supports robust tuning.

4) Comprehensive empirical validation. Experiments across multiple datasets and representative models show consistent and significant improvements over state-of-the-art denoising baselines, with lower computational and memory costs.

Our source code is available at https://github.com/Fammia/EARD.

2 Related Work

Denoising implicit feedback is essential for robust recommendation, as it often includes false positives (*e.g.*, accidental clicks) and false negatives (*e.g.*, unobserved preferences due to exposure bias). Early approaches used heuristic rules and statistical reweighting. **WRMF** [11] separates interactions into binary preferences and confidence scores based on frequency, optimizing with Alternating Least Squares for scalability and interpretability. **WBPR** [5] extends Bayesian Personalized Ranking by assigning popularity-aware sampling to negatives, mitigating sampling bias.

Later work leverages the memory effect in deep learning, where clean samples are learned earlier. **ADT** [21] uses truncated and reweighted losses to suppress high-loss samples. **IR** [26] iteratively refines labels through self-training. **SGDL** [6] adopts a two-phase approach that detects memorized samples, then adjusts loss weights via meta-learning and an LSTM scheduler. **DCF** [10] smooths loss histories and relabels samples to retain informative ones while filtering noise. **UDT** [2] models interactions as a two-stage ("willingness" to "action") Markov process and applies hierarchical loss weighting. **PLD** [30] resamples interactions using user-specific loss distributions that better separate clean from noisy data.

Recent methods address noise using structural signals, multi-view learning, and modality-specific features. **DRPN** [12] filters noise by comparing aggregated feedback. **DeCA** [24] detects noisy labels via cross-model disagreement. **RGCF** [19] prunes noisy edges in graph-based models using similarity scores and multi-view mutual information. **KRDN** [37] denoises knowledge graph triples and user-item links through attention masking and graph contrast. **RocSE** [29] improves robustness through structural filtering, embedding noise, and contrastive learning. **SLED** [35] uses local graph patterns in a two-stage unsupervised framework to estimate reliability. **DA-MRS** [28] handles multimodal noise via cross-modal consistency, noise-aware BPR loss, and contrastive objectives.

Beyond structural and loss-based methods, recent work explores advanced paradigms. **AutoDenoise** [16] applies reinforcement learning to select clean samples based on training utility. **BOD** [25] formulates sample weighting as bi-level optimization guided by performance. **DDRM** [36] uses diffusion models to generate embeddings from collaborative signals. **LLaRD** [20] leverages large language models and side information to infer preferences via chain-of-thought reasoning with information bottleneck regularization.

Recent research on robust recommendation has moved beyond isolated interactions by focusing on structural consistency and collaborative semantics [7, 17, 32]. Parallel efforts focus on improving efficiency by identifying and pruning redundant feature components [3, 14, 22, 23]. Meanwhile, adaptive mechanisms increasingly calibrate signal importance across heterogeneous contexts, emphasizing data reliability and quality [18, 31, 33, 34].

Building on insights from prior work, we present **EARD**, an entity-aware framework that integrates user and item reliability into interaction weighting. The design is lightweight, scalable, and model-agnostic, effectively addressing key limitations of existing methods, including the absence of entity-level modeling, high computational cost, and extensive hyperparameter tuning.

3 Method

3.1 Problem Formulation

3.1.1 Notation. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$ denote the sets of M users and N items. The set of observed user-item interactions is $O = \{(u, i) | \text{user } u \text{ interacted with item } i\}$. For training, we use the observed interactions O as positive instances and randomly sample a set of unobserved interactions $\mathcal{S}$ as negative instances, where $|\mathcal{S}| = k \cdot |O|$, where k is the negative sampling ratio. The full set of training instances for an epoch is denoted as $\mathcal{D} = O \cup \mathcal{S}$. Our goal is to learn a prediction function $\hat{y}_{ui} = f(u, i; \Theta)$ that minimizes a loss function, such as Binary Cross-Entropy: $\ell(\hat{y}_{ui}, y_{ui})$.

3.1.2 Denoising as a Weighted Optimization Problem. Recognizing the prevalence of noise, we formulate denoising as a weighted optimization problem. We associate a confidence weight $w_{ui} \in \mathbb{R}$ with each training instance $(u, i) \in \mathcal{D}$. The objective is to learn the model parameters Θ by minimizing the weighted loss over the set of training instances:

$$\min_{\Theta} \sum_{(u,i)\in\mathcal{D}} w_{ui} \cdot \ell(f(u, i; \Theta), y_{ui}) \tag{1}$$

3.2 Core Assumptions

Our method relies on two key observations: **1)** Noisy interactions tend to produce higher training losses, as supported by prior work in curriculum and robust learning [1, 8, 13] and illustrated in Figure 5a; **2)** Interaction reliability varies significantly across entities, as shown in Figure 1a and Appendix Figure 6. These insights motivate our entity-aware design, which weights interactions based on reliability estimated from average training loss.

3.3 The EARD Framework

Building on these assumptions, **EARD** assigns weights to user, item, and interaction factors based on their estimated noise levels, and integrates them into a unified final weight.

3.3.1 Factor-Driven Weighting Mechanism. As illustrated in Figure 2, **EARD** operates iteratively. At the end of each epoch t, it computes a weight w_{ui} for every instance (u, i) in the training set $\mathcal{D}^{(t)}$. The design is stateless and computationally efficient. For clarity, we use italic lowercase for scalar weights (w_{ui}), and bold lowercase for entity-level reliability score vectors ($\mathbf{w}_u, \mathbf{w}_i$).

Step 1: Base Weight Generation ($w_{ui,base}$). For each training instance $(u, i) \in \mathcal{D}^{(t)}$, we compute a base weight $w_{ui,\text{base}}$ to quantify its interaction-level factor. This is obtained by applying the Empirical Cumulative Distribution Function (ECDF) to the loss set, a non-parametric method that is robust to outliers. The base weight for instance (u, i) is defined as:

$$w_{ui,\text{base}}^{(t+1)} = \frac{1}{|\mathcal{L}^{(t)}|} \sum_{(u',i')\in\mathcal{L}^{(t)}} \mathbb{I}(-\ell_{u'i'}^{(t)} \le -\ell_{ui}^{(t)}), \tag{2}$$

where $\mathcal{L}^{(t)}$ is the collection of all losses from the training set $\mathcal{D}^{(t)}$ in epoch t. This process assigns higher weights to lower-loss samples without storing a dense matrix.

Step 2: Entity-Aware Weight Computation ($\mathbf{w}_u, \mathbf{w}_i$). We estimate user and item reliability using their average training loss in epoch t. For user u, the average loss is

$$\bar{\ell}_u^{(t)} = \frac{\sum_{i\in\mathcal{I}_u^{(t)}} \ell_{ui}^{(t)}}{|\mathcal{I}_u^{(t)}|}, \tag{3}$$

where $\mathcal{I}_u^{(t)}$ is the set of items interacted with by user u during epoch t. The same applies to items. Assuming lower average loss indicates higher reliability, we map these values to reputation scores using a rank-based linear function. Let $\text{rank}(\bar{\ell}_u^{(t)})$ denote the ascending rank of user u's average loss. Then the user reputation factor is

$$w_u^{(t+1)} = \alpha + (\beta - \alpha) \cdot \frac{\text{rank}(\bar{\ell}_u^{(t)}) - 1}{M - 1}, \tag{4}$$

where $\alpha, \beta \in [0, \infty)$ are hyperparameters and $\alpha \le \beta$. This mapping assigns the highest score β to the most reliable entity (lowest loss) and α to the least. We repeat this process for all users and items to obtain reliability score vectors $\mathbf{w}_u^{(t+1)} \in \mathbb{R}^M$ and $\mathbf{w}_i^{(t+1)} \in \mathbb{R}^N$, which are then used for the final weight computation.

Step 3: Final Weight Fusion (w_{ui}). We obtain the final weight $w_{ui}^{(t+1)}$ for each training instance $(u, i) \in \mathcal{D}^{(t+1)}$ by fusing its

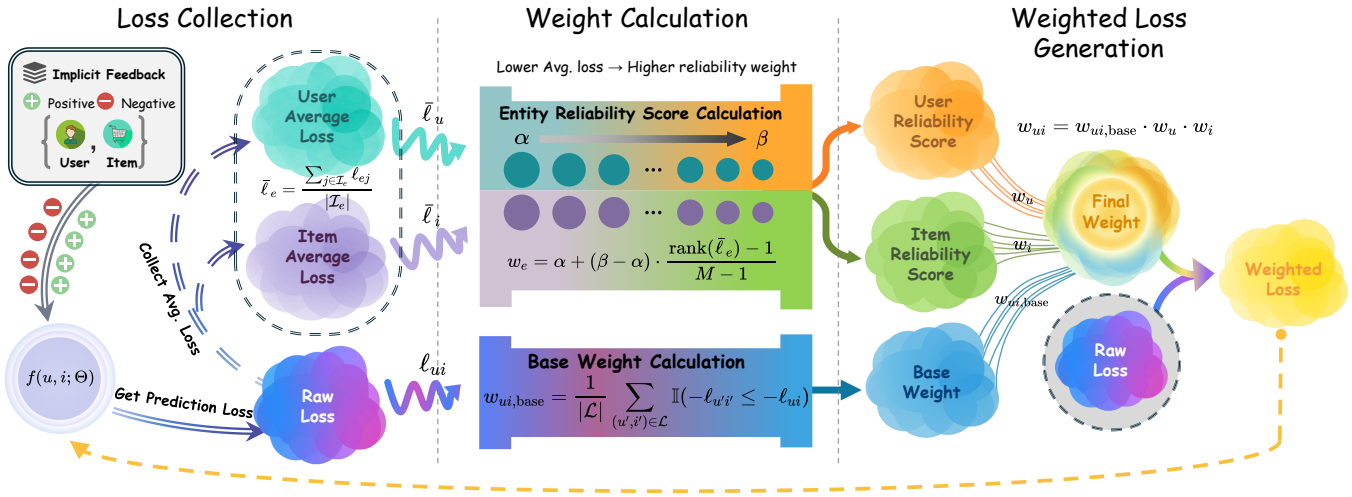

Figure 2: Overview of EARD. At the end of each epoch, the framework performs three steps: (1) *Collect Loss*: gather losses for all interactions and compute average loss per entity; (2) *Entity Reliability and Base Weight Calculation*: estimate entity reliability via a linear mapping of average loss from α to β, and assign base weights using the ECDF of negative losses; (3) *Weight Fusion*: combine interaction and entity weights through element-wise multiplication to generate training weights for the next epoch.

interaction-level confidence and the reliability of its associated entities. This is performed via a simple scalar multiplication:

$$w_{ui}^{(t+1)} = w_{ui,\text{base}}^{(t+1)} \cdot w_u^{(t+1)} \cdot w_i^{(t+1)} \tag{5}$$

This formulation ensures high confidence only when both the interaction and its associated entities are considered reliable.

3.4 Complexity Analysis

We analyze the computational complexity of EARD to demonstrate its efficiency and scalability. Let $|\mathcal{D}| = |\mathcal{O}| + |\mathcal{S}|$ denote the total number of training samples per epoch.

Time Complexity. EARD introduces three additional operations per epoch: (1) computing average losses in $O(|\mathcal{D}|)$; (2) generating reliability scores by sorting, costing $O(M \log M + N \log N)$; and (3) computing ECDF-based weights, requiring $O(|\mathcal{D}| \log |\mathcal{D}|)$.

Space Complexity. The method maintains the loss and weight for all samples (i.e., valid interactions) within each training epoch, leading to a space complexity of $O(|\mathcal{D}|)$.

4 Experiments

4.1 Experimental Setup

We empirically evaluate the effectiveness and efficiency of **EARD** through the following research questions:

1) RQ1 (Overall Performance): Does EARD consistently outperform SOTA denoising methods across models and datasets?
2) RQ2 (Ablation Study): How do the entity-level and interaction-level weights contribute to overall performance?
3) RQ3 (Hyperparameter Sensitivity): How sensitive is EARD to its hyperparameters α and β, and is it easy to tune?
4) RQ4 (Mechanism Insight): Can EARD effectively distinguish true-positive (clean) and false-positive (noisy) interactions based on the learned interaction and entity weights?

Table 1: Dataset statistics. "#False Positives" denotes noisy interactions (ratings ≤ 3) addressed by our framework.

Dataset	#Users	#Items	#Interactions	#False Positives	Sparsity
Yelp	45,548	57,396	1,672,520	260,581	0.0640%
Amazon-book	80,464	98,663	2,714,021	199,475	0.0342%
ML-1M	6,040	3,953	706,391	338,746	2.959%

5) RQ5 (Efficiency): What is the computational overhead of EARD, and is it scalable to large datasets? We report peak memory usage and training epochs until convergence, comparing with vanilla training and other denoising methods.

4.1.1 Datasets. We evaluate EARD on three public datasets: **Yelp**[1], **Amazon-Book**[2], and **ML-1M**[3]. Following [21], we binarize interactions by treating ratings ≤ 3 as *false positives* (noisy) and > 3 as *true positives*. **Yelp** contains user reviews for local businesses. **Amazon-Book** consists of user ratings for books. **ML-1M** includes movie ratings from the GroupLens research group.

All datasets are randomly split into training, validation, and test sets in an **8:1:1** ratio. Key statistics are summarized in Table 1.

4.1.2 Backbone Models. To demonstrate the model-agnostic nature of EARD, we apply it to three representative collaborative filtering models: **GMF** [9], which uses element-wise multiplication of user and item embeddings; **NeuMF** [9], which combines GMF with multilayer perceptrons to model both linear and non-linear interactions; and **CDAE** [27], an autoencoder that reconstructs user interaction vectors from corrupted inputs.

[1] https://www.yelp.com/dataset
[2] https://jmcauley.ucsd.edu/data/amazon/
[3] https://grouplens.org/datasets/movielens/1m/

Table 2: Overall performance of three representative backbone trained using seven awesome different denoising methods. 'Recall@K' and 'NDCG@K' are shortened as 'R@K' and 'N@K'. 'Δ%' refers to the relative improvement of our method compared to the other baselines. '*' indicates statistically significant improvement by t-test ($p < 0.05$) to other methods.

Model	Method	ML-1M				Yelp				Amazon-book			
		R@50	R@100	N@50	N@100	R@50	R@100	N@50	N@100	R@50	R@100	N@50	N@100
GMF	Base	0.3967	0.5523	0.3375	0.3970	0.1020	0.1673	0.0406	0.0552	0.0851	0.1347	0.0341	0.0450
	WRMF	0.3785	0.5278	0.3286	0.3849	0.0875	0.1427	0.0363	0.0487	0.0734	0.1157	0.0301	0.0394
	R-CE	0.3749	0.5240	0.3277	0.3840	0.0864	0.1365	0.0367	0.0481	0.0657	0.1041	0.0268	0.0353
	T-CE	0.3686	0.5226	0.3154	0.3739	0.0900	0.1471	0.0363	0.0493	0.0738	0.1147	0.0306	0.0397
	DeCA	0.2757	0.4007	0.2294	0.2778	0.0466	0.0823	0.0169	0.0248	0.0616	0.1050	0.0220	0.0314
	BOD	0.4030	0.5569	0.3427	0.4014	0.0954	0.1559	0.0387	0.0522	0.0762	0.1223	0.0302	0.0404
	PLD	0.4121	0.5716	0.3517	0.4128	0.0982	0.1610	0.0394	0.0535	0.0779	0.1251	0.0310	0.0414
	UDT	0.4131	0.5585	0.3426	0.3987	0.1352	0.2118	0.0569	0.0741	**0.1228**	**0.1807**	**0.0524**	**0.0653**
	Ours	**0.4342***	**0.5851***	**0.3667***	**0.4241***	**0.1375***	**0.2169***	**0.0575***	**0.0753***	0.1193	0.1765	0.0506	0.0633
	Δ%	5.11%	2.36%	4.26%	2.74%	1.70%	2.41%	1.05%	1.62%	-2.85%	-2.32%	-3.44%	-3.06%
NeuMF	Base	0.3868	0.5431	0.3171	0.3770	0.0875	0.1516	0.0332	0.0475	0.0844	0.1341	0.0330	0.0439
	WRMF	0.3959	0.5495	0.3300	0.3889	0.0806	0.1353	0.0318	0.0441	0.0711	0.1141	0.0280	0.0374
	R-CE	0.3803	0.5367	0.3188	0.3782	0.0803	0.1306	0.0325	0.0438	0.0606	0.0970	0.0244	0.0324
	T-CE	0.3943	0.5533	0.3205	0.3820	0.0831	0.1391	0.0321	0.0447	0.0693	0.1106	0.0274	0.0365
	DeCA	0.2420	0.3510	0.2109	0.2516	0.0464	0.0904	0.0162	0.0258	0.0420	0.0792	0.0144	0.0222
	BOD	0.3782	0.5242	0.3262	0.3816	0.0851	0.1371	0.0349	0.0467	0.0768	0.1262	0.0287	0.0396
	PLD	0.3847	0.5339	0.3345	0.3920	0.0866	0.1398	0.0357	0.0482	0.0791	0.1293	0.0295	0.0406
	UDT	0.4126	0.5649	0.3314	0.3901	0.1085	0.1790	0.0422	0.0580	0.1091	0.1650	0.0459	0.0583
	Ours	**0.4257***	**0.5747***	**0.3617***	**0.4181***	**0.1306***	**0.2057***	**0.0536***	**0.0704***	**0.1235***	**0.1820***	**0.0530***	**0.0658***
	Δ%	3.17%	1.73%	8.13%	6.66%	20.37%	14.92%	27.01%	21.38%	13.20%	10.30%	15.47%	12.86%
CDAE	Base	0.4299	0.5802	0.3656	0.4252	0.1165	0.1877	0.0466	0.0625	0.1024	0.1574	0.0417	0.0538
	WRMF	0.4095	0.5590	0.3505	0.4081	0.0943	0.1534	0.0375	0.0507	0.0896	0.1372	0.0377	0.0438
	R-CE	0.4114	0.5600	0.3565	0.4135	0.1161	0.1801	0.0488	0.0632	0.1022	0.1560	0.0424	0.0542
	T-CE	0.4033	0.5572	0.3394	0.3994	0.1165	0.1806	0.0504	0.0652	0.1088	0.1645	0.0454	0.0575
	DeCA	0.2446	0.3293	0.1763	0.2123	0.0767	0.1212	0.0306	0.0409	0.0663	0.1040	0.0265	0.0343
	BOD	0.4161	0.5668	0.3542	0.4125	0.1115	0.1755	0.0457	0.0601	0.0911	0.1409	0.0374	0.0482
	PLD	0.4296	0.5836	0.3620	0.4200	0.1144	0.1804	0.0468	0.0618	0.0932	0.1441	0.0383	0.0492
	UDT	0.4347	0.5818	0.3749	0.4317	0.1321	0.2013	0.0566	0.0722	0.1219	0.1768	0.0525	0.0647
	Ours	**0.4500***	**0.5930***	**0.3900***	**0.4447***	**0.1562***	**0.2346***	**0.0675***	**0.0850***	**0.1447***	**0.2073***	**0.0633***	**0.0770***
	Δ%	3.52%	1.61%	4.03%	3.01%	18.24%	16.54%	19.26%	17.73%	18.70%	17.25%	20.57%	19.01%

4.1.3 Evaluation Protocol. We follow standard practice by ranking all non-interacted items for each user. Evaluation focuses on *true-positive* test set, reflecting the goal of recommending items users genuinely prefer. We report two common top-K ranking metrics: **Recall@K** and **NDCG@K**, with K={50, 100}.

4.1.4 Baselines. We compared EARD against backbone models and representative denoising baselines, including WRMF [11], R-CE [21], T-CE [21], model-disagreement approaches like DeCA [24], BOD [25], UDT [2] and PLD [30]. [4]

4.1.5 Implementation Details. All experiments were conducted on an NVIDIA 4090 D GPU using the Adam optimizer [15]. The embedding size was 32, the learning rate 10^{-3}, the batch size 2048, and the negative sampling rate (k) 1. For EARD, hyperparameters α and β were tuned within $[0, 5]$, with optimal settings reported in Table 6 (Appendix A). All results are averaged over five independent runs to ensure stability and reliability.

[4]We exclude LLaRD [20] because it depends on additional side information (e.g., item descriptions, user reviews, and external world knowledge from large language models), which is unavailable to other baselines and inconsistent with our interaction-only setting. SGDL [6] is also omitted, as it does not finish training within a reasonable time on datasets exceeding one million interactions.

4.2 Performance Comparison (RQ1)

We benchmarked **EARD** against state-of-the-art baselines across all datasets and backbone models, using their best reported configurations for a fair comparison.

As shown in Table 2, EARD consistently outperforms all baselines. Heuristic methods such as R-CE and T-CE yield unstable results and occasionally underperform the backbone, indicating that naively pruning high-loss samples can discard informative but challenging instances. More complex approaches like DeCA also underperform, suggesting that added architectural complexity does not guarantee improved denoising. In contrast, EARD achieves robust gains by leveraging entity-level attribution, with particularly strong improvements on larger datasets where performance gaps are more pronounced. Notably, even on CDAE, which is inherently robust to noise, EARD delivers significant improvements (e.g., +19.26% and +20.57% NDCG@50 on Yelp and Amazon-Book), demonstrating its complementarity to model-based robustness.

Compared to UDT, EARD generally achieves superior performance, especially on advanced models like NeuMF and CDAE. The only exception is GMF on Amazon-Book, where UDT slightly outperforms it. On NeuMF, EARD delivers an average NDCG@50

Table 3: Ablation results on ML-1M, Yelp, and Amazon-Book for GMF, NeuMF, and CDAE. We evaluate three components: ECDF-based Weight (BW), Item Weight (IF), and User Weight (UF). ✓indicates active, × indicates inactive.

Model	Components			ML-1M				Yelp				Amazon-book			
	BW	IF	UF	R@50	R@100	N@50	N@100	R@50	R@100	N@50	N@100	R@50	R@100	N@50	N@100
GMF	×	×	×	0.3967	0.5523	0.3375	0.3970	0.1020	0.1673	0.0406	0.0552	0.0851	0.1347	0.0341	0.0450
	✓	×	×	0.4271	0.5765	0.3593	0.4166	0.1373	0.2158	0.0575	0.0750	0.1143	0.1745	0.0471	0.0604
	✓	✓	×	**0.4353**	0.5821	**0.3681**	**0.4242**	0.1373	0.2164	0.0574	0.0751	0.1190	0.1795	0.0498	0.0631
	✓	×	✓	0.4286	0.5813	0.3636	0.4218	**0.1380**	0.2163	**0.0577**	0.0752	**0.1195**	**0.1806**	0.0499	0.0633
	✓	✓	✓	0.4342	**0.5851**	0.3667	0.4241	0.1375	**0.2169**	0.0575	**0.0753**	0.1193	0.1765	**0.0506**	**0.0633**
NeuMF	×	×	×	0.3868	0.5431	0.3171	0.3770	0.0875	0.1516	0.0332	0.0475	0.0844	0.1341	0.0330	0.0439
	✓	×	×	0.4226	0.5740	0.3605	0.4176	0.1040	0.1713	0.0410	0.0516	0.0912	0.1445	0.0361	0.0479
	✓	✓	×	0.4252	0.5739	**0.3623**	**0.4184**	0.1238	0.1984	0.0494	0.0660	0.1104	0.1677	0.0453	0.0578
	✓	×	✓	0.4225	0.5738	0.3595	0.4166	0.1203	0.1959	0.0480	0.0649	0.1086	0.1676	0.0446	0.0576
	✓	✓	✓	**0.4257**	**0.5747**	0.3617	0.4181	**0.1306**	**0.2057**	**0.0536**	**0.0704**	**0.1235**	**0.1820**	**0.0530**	**0.0658**
CDAE	×	×	×	0.4299	0.5802	0.3656	0.4252	0.1165	0.1877	0.0466	0.0625	0.1024	0.1574	0.0417	0.0538
	✓	×	×	0.4390	0.5871	0.3794	0.4365	0.1509	0.2299	0.0636	0.0812	0.1389	0.2030	0.0592	0.0731
	✓	✓	×	0.4405	0.5839	0.3855	0.4404	0.1535	0.2318	0.0652	0.0827	0.1426	0.2055	0.0619	0.0756
	✓	×	✓	0.4447	0.5918	0.3820	0.4390	**0.1573**	**0.2371**	0.0664	0.0841	**0.1458**	**0.2101**	0.0633	**0.0773**
	✓	✓	✓	**0.4500**	**0.5930**	**0.3900**	**0.4447**	0.1562	0.2346	**0.0675**	**0.0850**	0.1447	0.2073	**0.0633**	0.0770

improvement of up to **27.01%** on Yelp. It also requires fewer hyperparameters (Ours: 2 vs. UDT: 4), making it more efficient to tune and better suited for practical deployment. These results highlight the advantage of entity-level modeling for denoising implicit feedback.

4.3 Ablation Study (RQ2)

4.3.1 Component Contribution Analysis. We conducted an ablation study (Table 3) to evaluate the contribution of each component in EARD, which comprises three elements: the interaction-specific Base Weight (**BW**), Item Factor (**IF**), and User Factor (**UF**). Each component was selectively disabled, and performance was measured using Recall@50/100 and NDCG@50/100.

The Foundational Role of Interaction-level Weighting (BW). Enabling only BW (✓,×,×) yields a clear and substantial improvement over the vanilla baseline (×,×,×) across all models and datasets. For instance, on NeuMF with the ML-1M dataset, relying solely on BW increases NDCG@50 from 0.3171 to 0.3605, a relative gain of **13.7%**. This result strongly substantiates our core hypothesis: an interaction's training loss serves as a powerful and reliable proxy for its underlying quality and can be directly exploited for effective denoising, forming the foundation of our framework.

Complementary and Context-Dependent Effects of Entity Factors (IF & UF). Introducing either the Item Factor or the User Factor on top of BW almost always brings further performance gains, demonstrating the critical value of entity-aware modeling. The effects of IF and UF appear to be complementary and context-dependent. For example, the **User Factor** (✓,×,✓) shows a particularly strong impact on the CDAE model with the Amazon-Book dataset, suggesting that when user behaviors are highly diverse and noisy, explicitly modeling user-specific reliability is crucial. Conversely, the **Item Factor** (✓,✓,×) achieves the best performance on three out of four metrics for GMF on ML-1M, indicating that in scenarios where item-side noise (e.g., clickbait or low-quality items) is dominant, identifying unreliable items is a primary driver of improvement.

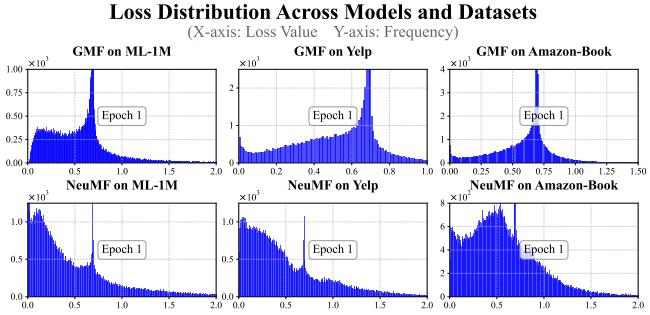

Figure 3: Training loss distributions at the end of the first epoch across different models and datasets.

Synergy in Full Configuration and Its Implications. The full EARD configuration (✓,✓,✓) generally delivers top-tier performance, highlighting the strong synergy among its components. BW provides a solid foundation for interaction quality, while IF and UF contribute personalized and orthogonal signals that enhance reliability. For instance, on the Yelp dataset with NeuMF, the full model achieves an NDCG@50 of 0.0536, outperforming BW-only by 30.7%, BW+IF by 8.5%, and BW+UF by 11.7%. These results demonstrate that modeling all three factors together enables EARD to generalize effectively in sparse and noisy environments.

4.3.2 Exploration of Base Weight Calculation Methods. The interaction specific **Base Weight (BW)** is critical to EARD. As shown in Figure 3, loss distributions vary across models and datasets and often exhibit long-tail outliers. This underscores the need for a non-parametric weighting method that adapts to diverse distribution shapes while reducing the influence of extreme values. ECDF is distribution-agnostic and robust to outliers, as it depends on relative rank rather than absolute magnitude. Moreover, it produces smooth weights in the $[0, 1]$ range, avoiding information loss from hard thresholds and enabling stable denoising.

Table 4: Full evaluation of different base weighting strategies (Uniform, GMM, Top-k Selection, Linear Scaling, ECDF) integrated into the EARD framework, across three datasets and three models. Metrics include Recall@50/100 and NDCG@50/100. ECDF consistently achieves the highest performance, validating its effectiveness and robustness.

Model	Method	ML-1M				Yelp				Amazon-Book			
		R@50	R@100	N@50	N@100	R@50	R@100	N@50	N@100	R@50	R@100	N@50	N@100
GMF	Uniform	0.3844	0.5393	0.3219	0.3815	0.1028	0.1679	0.0410	0.0555	0.1136	0.1743	0.0474	0.0608
	Top-k Selection	0.4011	0.5668	0.3161	0.3806	0.1266	0.1995	0.0530	0.0695	0.0980	0.1486	0.0415	0.0529
	Linear Scaling	0.3873	0.5421	0.3244	0.3889	0.1026	0.1682	0.0409	0.0556	0.1138	0.1742	0.0475	0.0608
	GMM	0.3215	0.4501	0.2853	0.3337	0.0879	0.1389	0.0367	0.0481	0.0389	0.0629	0.0157	0.0210
	ECDF (Ours)	**0.4342**	**0.5851**	**0.3667**	**0.4241**	**0.1375**	**0.2169**	**0.0575**	**0.0753**	**0.1193**	**0.1765**	**0.0506**	**0.0633**
NeuMF	Uniform	0.4005	0.5510	0.3374	0.3944	0.1173	0.1874	0.0470	0.0626	0.1059	0.1604	0.0440	0.0559
	Top-k Selection	0.4073	0.5629	0.3273	0.3868	0.1230	0.1930	0.0511	0.0669	0.1064	0.1567	0.0460	0.0573
	Linear Scaling	0.4023	0.5541	0.3396	0.3971	0.1190	0.1914	0.0482	0.0644	0.1103	0.1682	0.0459	0.0585
	GMM	0.3080	0.4364	0.2669	0.3164	0.0427	0.0701	0.0179	0.0241	0.0374	0.0612	0.0159	0.0213
	ECDF (Ours)	**0.4257**	**0.5747**	**0.3617**	**0.4181**	**0.1306**	**0.2057**	**0.0536**	**0.0704**	**0.1235**	**0.1820**	**0.0530**	**0.0658**
CDAE	Uniform	0.4299	0.5779	0.3719	0.4292	0.1426	0.2190	0.0611	0.0781	0.1256	0.1853	0.0537	0.0667
	Top-k Selection	0.4349	0.5780	0.3682	0.4238	0.1470	0.2213	0.0633	0.0800	0.1290	0.1849	0.0572	0.0696
	Linear Scaling	0.4323	0.5805	0.3736	0.4309	0.1430	0.2192	0.0613	0.0783	0.1249	0.1864	0.0533	0.0667
	GMM	0.3319	0.4684	0.2836	0.3371	0.1239	0.1864	0.0541	0.0682	0.1178	0.1687	0.0518	0.0629
	ECDF	**0.4500**	**0.5930**	**0.3900**	**0.4447**	**0.1562**	**0.2346**	**0.0675**	**0.0850**	**0.1447**	**0.2073**	**0.0633**	**0.0770**

To validate effectiveness, we compare ECDF with four alternatives: (1) **Uniform**, assigning a constant weight of 1 to all samples; (2) **GMM**, fitting a Gaussian Mixture Model to the loss distribution; (3) **Top-k Selection**, retaining the k lowest-loss samples per batch; and (4) **Linear Scaling**, applying a linear transformation to the losses. Each method replaces only the ECDF component in the EARD framework. Pseudocode is provided in Appendix F.

As shown in Table 4, ECDF consistently outperforms all alternative weighting strategies, confirming its effectiveness and robustness. The GMM-based method performs poorly due to its reliance on Gaussian assumptions, which fail under the diverse and non-standard loss distributions. Top-k Selection applies a hard threshold, discarding high-loss samples entirely, which can eliminate informative but difficult interactions and leads to unstable training behavior. Linear Scaling directly maps loss magnitudes to weights, making it highly sensitive to outliers; a few large losses can dominate the weight range and distort the learning dynamics. In contrast, ECDF relies on loss rank rather than magnitude, making it robust to outliers and distribution-agnostic. Its smooth weighting over $[0, 1]$ preserves fine-grained differences without abrupt cutoffs, making it well-suited for denoising implicit feedback.

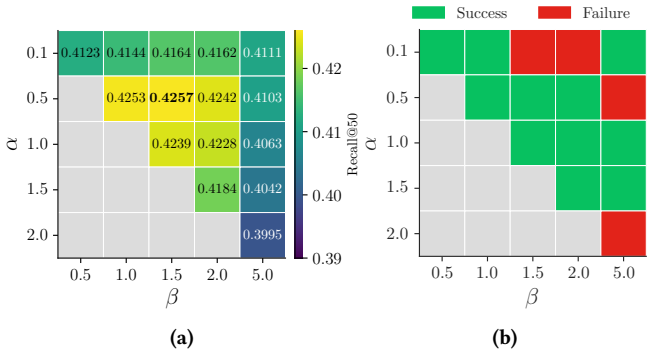

Figure 4: (a) Recall@50 performance landscape over $[\alpha, \beta]$ using NeuMF on ML-1M. (b) Hessian-based concavity test at 15 representative points from (a).

4.4 Hyperparameter Sensitivity Analysis (RQ3)

We evaluated the sensitivity of EARD to its two key hyperparameters, α and β, which define the bounds of entity weights. A robust model should retain strong performance under small hyperparameter variations. As shown in Figure 4, the Recall@50 landscape for NeuMF on ML-1M is smooth and unimodal, with a broad region of high performance. This pattern is consistent across all backbones and datasets, as illustrated in Figures 9 and 10 in the appendix.

To quantitatively support the visual observation of a smooth, unimodal landscape, we performed a local curvature analysis using a discrete Hessian-based concavity test (Algorithm 3 in Appendix G). The results, shown in Figure 4b, reveal a strong correlation between mathematically verified concave regions and the high-performance areas of the landscape. This alignment confirms that the objective surface exhibits a favorable and well-structured geometry, devoid of erratic local optima. Such a "hill-shaped" topology is particularly valuable, as it suggests that EARD is robust to hyperparameter selection; small perturbations near the optimum do not lead to significant performance degradation. As a result, the framework remains easy to tune in practice.

4.5 Denoising Mechanism Analysis (RQ4)

We analyzed training loss and assigned weights for two groups: true positives (TP), representing clean interactions, and false positives (FP), representing noisy ones. As shown in Figure 5a, FP samples generally exhibit higher average losses than TP samples, supporting the small-loss assumption and suggesting that noisy interactions are harder to fit. In contrast, the vanilla model eventually reduces both TP and FP losses toward zero, indicating that it overfits the noise in the absence of denoising.

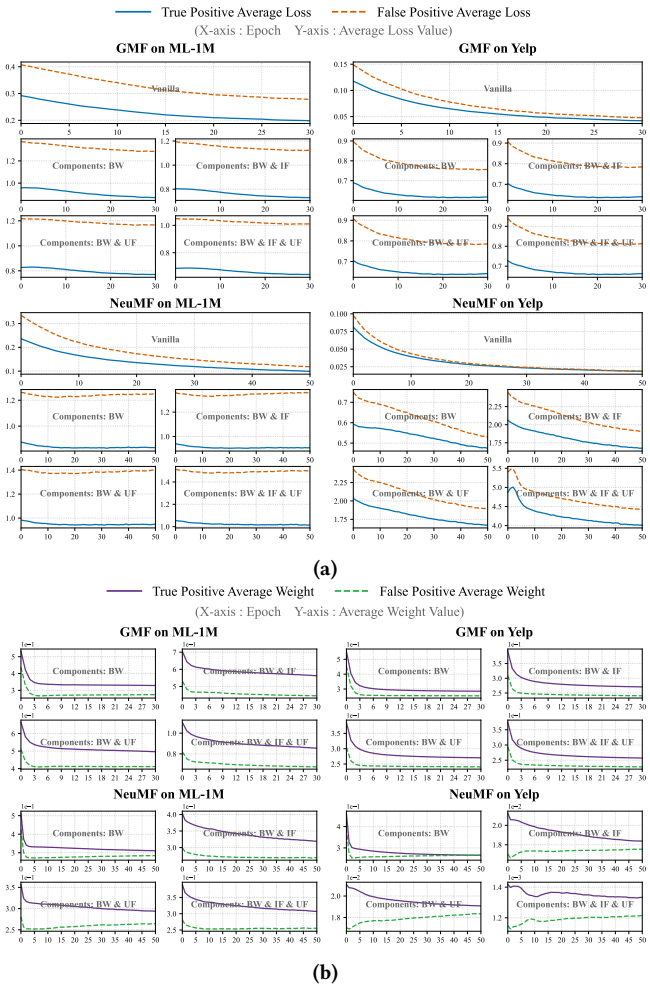

Figure 5: Training dynamics of TP and FP samples across models and datasets. Base Weight (BW) is interaction-specific, while User Factor (UF) and Item Factor (IF) model user and item reliability. (a) TP and FP losses over epochs. EARD maintains a clear loss gap, with FP samples exhibiting higher losses, supporting the small-loss assumption. (b) Sample weights for TP and FP. EARD consistently assigns higher weights to TP samples, demonstrating its effectiveness in identifying clean interactions and suppressing noise.

EARD amplifies the separation between true positive (TP) and false positive (FP) samples during training. As shown in Figure 5b, models with our framework maintain a clear and stable gap in assigned weights between TP and FP samples across architectures (GMF and NeuMF) and datasets (ML-1M and Yelp). This separation holds across all component settings (e.g., BW, BW & IF, BW & UF), with TP samples generally receiving higher weights, especially in early training. For example, on GMF with ML-1M, TP weights quickly converge near 10^{-1}, while FP weights stay around 10^{-3}. These results confirm that our reweighting mechanism effectively distinguishes clean from noisy data and guides learning accordingly.

Table 5: Computational efficiency comparison of denoising methods. Each cell reports peak GPU memory usage (MB, Mem) and training epochs to convergence (Ep).

Method	GMF						NeuMF					
	ML-1M		Yelp		Amazon-Book		ML-1M		Yelp		Amazon-Book	
	Mem	Ep	Mem	Ep	Mem	Ep	Mem	Ep	Mem	Ep	Mem	Ep
Vanilla	528	46	1232	172	1894	131	958	36	6476	91	10890	89
DeCA	1510	63	4050	218	7346	182	2972	49	25347	123	34325	132
BOD	1140	51	2713	153	4621	144	2130	41	14793	84	28641	101
Ours	559	48	1330	106	2124	188	1063	32	7382	77	11870	96

4.6 Efficiency Analysis (RQ5)

To address **RQ5**, we compare the overhead of **EARD** with the vanilla pipeline and other denoising methods using peak GPU memory usage and training epochs to convergence. Epochs are reported instead of wall-clock time to ensure fair evaluation across different server conditions. Table 5 summarizes the results, using per-epoch validation and early stopping after 10 non-improving epochs.

The results demonstrate that EARD offers a clear efficiency advantage. Although it introduces a modest and predictable memory overhead of 5 to 15 percent compared to the vanilla baseline, methods such as **DeCA** and **BOD** consume substantially more resources, requiring up to **3x** and **2x** more memory, respectively. This indicates that EARD achieves performance improvements without incurring the high memory costs. Regarding convergence speed, while advanced denoising methods, including ours, DeCA, and BOD, occasionally require more epochs than vanilla training, we attribute this to a more thorough optimization process that avoids premature convergence and yields better final performance.

In summary, EARD achieves a favorable balance between performance and efficiency. It delivers state-of-the-art results (Table 2) with a minimal memory footprint compared to strong baselines. Its lightweight and highly parallelizable design makes it practical for large-scale industrial recommender systems.

5 Conclusion

We proposed EARD, a lightweight and effective framework for denoising implicit feedback by jointly modeling both entity and interaction level reliability. By leveraging average loss as a proxy for noise, EARD captures user- and item-specific noise patterns and assigns fine-grained confidence scores through a nonparametric ECDF-based weighting scheme. Extensive experiments across diverse datasets and backbone models show that EARD consistently outperforms state-of-the-art denoising baselines, while incurring lower computational cost and requiring minimal hyperparameter tuning. These results underscore the importance of entity-aware modeling and establish EARD as a robust, scalable, and practical solution for recommendation in noisy real-world environments.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (Grant Nos. 62406303 and U25B2072), the Anhui Province Science and Technology Innovation Project (Grant No. 202423k09020010), and the Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education of China (Grant No. BigKEOpen2025-02).

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

[2] Haoyan Chua, Yingpeng Du, Zhu Sun, Ziyan Wang, Jie Zhang, and Yew-Soon Ong. 2024. Unified Denoising Training for Recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 612–621.

[3] Zhaocheng Du, Chuhan Wu, Qinglin Jia, Jieming Zhu, and Xu Chen. 2024. A Tutorial on Feature Interpretation in Recommender Systems. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 1281–1282.

[4] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM conference on recommender systems*. 101–109.

[5] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Personalized ranking for non-uniformly sampled items. In *Proceedings of KDD cup 2011*. PMLR, 231–247.

[6] Yunjun Gao, Yuntao Du, Yujia Hu, Lu Chen, Xinjun Zhu, Ziquan Fang, and Baihua Zheng. 2022. Self-guided learning to denoise for robust recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1412–1422.

[7] Zhong Guan, Likang Wu, Hongke Zhao, Ming He, and Jianpin Fan. 2025. Enhancing collaborative semantics of language model-driven recommendations via graph-aware learning. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[8] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems* 31 (2018).

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[10] Zhuangzhuang He, Yifan Wang, Yonghui Yang, Peijie Sun, Le Wu, Haoyue Bai, Jinqi Gong, Richang Hong, and Min Zhang. 2024. Double correction framework for denoising recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1062–1072.

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*. Ieee, 263–272.

[12] Yunfan Hu, Zhaopeng Qiu, and Xian Wu. 2022. Denoising neural network for news recommendation with positive and negative implicit feedback. *arXiv preprint arXiv:2204.04397* (2022).

[13] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*. PMLR, 2304–2313.

[14] Zhonghua Jiang, Kui Chen, Kunxi Li, Keting Yin, Yiyun Zhou, Zhaode Wang, Chengfei Lv, and Shengyu Zhang. 2025. AccKV: Towards Efficient Audio-Video LLMs Inference via Adaptive-Focusing and Cross-Calibration KV Cache Optimization. *arXiv preprint arXiv:2511.11106* (2025).

[15] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

[16] Weilin Lin, Xiangyu Zhao, Yejing Wang, Yuanshao Zhu, and Wanyu Wang. 2023. Autodenoise: Automatic data instance denoising for recommendations. In *Proceedings of the ACM Web Conference 2023*. 1003–1011.

[17] Fei Liu, Xuegang Hu, Shuochen Liu, Chenyang Bu, and Le Wu. 2023. Meta multi-agent exercise recommendation: A game application perspective. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 1441–1452.

[18] Chuan Qin, Xin Chen, Chengrui Wang, Pengmin Wu, Xi Chen, Yihang Cheng, Jingyi Zhao, Meng Xiao, Xiangchao Dong, Qingqing Long, et al. 2025. Scihorizon: Benchmarking ai-for-science readiness from scientific data to large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 5754–5765.

[19] Changxin Tian, Yuexiang Xie, Yaliang Li, Nan Yang, and Wayne Xin Zhao. 2022. Learning to denoise unreliable interactions for graph collaborative filtering. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 122–132.

[20] Shuyao Wang, Zhi Zheng, Yongduo Sui, and Hui Xiong. 2025. Unleashing the Power of Large Language Model for Denoising Recommendation. In *Proceedings of the ACM on Web Conference 2025*. 252–263.

[21] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. In *Proceedings of the 14th ACM international conference on web search and data mining*. 373–381.

[22] Xianquan Wang, Zhaocheng Du, Jieming Zhu, Chuhan Wu, Qinglin Jia, and Zhenhua Dong. 2025. TayFCS: Towards Light Feature Combination Selection for Deep Recommender Systems. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 5007–5017.

[23] Xianquan Wang, Likang Wu, Zhi Li, Haitao Yuan, Shuanghong Shen, Huibo Xu, Yu Su, and Chenyi Lei. 2025. Mitigating redundancy in deep recommender systems: A field importance distribution perspective. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 1515–1526.

[24] Yu Wang, Xin Xin, Zaiqiao Meng, Joemon M Jose, Fuli Feng, and Xiangnan He. 2022. Learning robust recommenders through cross-model agreement. In *Proceedings of the ACM web conference 2022*. 2015–2025.

[25] Zongwei Wang, Min Gao, Wentao Li, Junliang Yu, Linxin Guo, and Hongzhi Yin. 2023. Efficient bi-level optimization for recommendation denoising. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 2502–2511.

[26] Zitai Wang, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2021. Implicit feedbacks are not always favorable: Iterative relabeled one-class collaborative filtering against noisy interactions. In *Proceedings of the 29th ACM international conference on multimedia*. 3070–3078.

[27] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*. 153–162.

[28] Guipeng Xv, Xinyu Li, Ruobing Xie, Chen Lin, Chong Liu, Feng Xia, Zhanhui Kang, and Leyu Lin. 2024. Improving multi-modal recommender systems by denoising and aligning multi-modal content and user feedback. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3645–3656.

[29] Haibo Ye, Xinjie Li, Yuan Yao, and Hanghang Tong. 2023. Towards robust neural graph collaborative filtering via structure denoising and embedding perturbation. *ACM Transactions on Information Systems* 41, 3 (2023), 1–28.

[30] Kaike Zhang, Qi Cao, Yunfan Wu, Fei Sun, Huawei Shen, and Xueqi Cheng. 2025. Personalized Denoising Implicit Feedback for Robust Recommender System. In *Proceedings of the ACM on Web Conference 2025*. 4470–4481.

[31] Kai Zhang, Qi Liu, Hao Qian, Biao Xiang, Qing Cui, Jun Zhou, and Enhong Chen. 2021. Eatn: An efficient adaptive transfer network for aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 377–389.

[32] Kai Zhang, Zhihong Pan, Lei Yu, Qi Liu, Hongke Zhao, Chaochao Chen, and Enhong Chen. 2025. SimCDR: Preserving Intra-Domain Similarities of Users for Cross-Domain Recommendation. *ACM Transactions on Information Systems* (2025).

[33] Kai Zhang, Hao Qian, Qing Cui, Qi Liu, Longfei Li, Jun Zhou, Jianhui Ma, and Enhong Chen. 2021. Multi-interactive attention network for fine-grained feature learning in ctr prediction. In *Proceedings of the 14th ACM international conference on web search and data mining*. 984–992.

[34] Lei Zhang, Wuji Zhang, Likang Wu, and Hongke Zhao. 2025. GCTN: Graph Competitive Transfer Network for Cross-Domain Multi-Behavior Prediction. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[35] Shengyu Zhang, Tan Jiang, Kun Kuang, Fuli Feng, Jin Yu, Jianxin Ma, Zhou Zhao, Jianke Zhu, Hongxia Yang, Tat-Seng Chua, et al. 2023. Sled: Structure learning based denoising for recommendation. *ACM Transactions on Information Systems* 42, 2 (2023), 1–31.

[36] Jujia Zhao, Wang Wenjie, Yiyan Xu, Teng Sun, Fuli Feng, and Tat-Seng Chua. 2024. Denoising diffusion recommender model. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1370–1379.

[37] Xinjun Zhu, Yuntao Du, Yuren Mao, Lu Chen, Yujia Hu, and Yunjun Gao. 2023. Knowledge-refined denoising network for robust recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*. 362–371.

A Hyperparameter Settings

We report the optimal hyperparameter settings for the proposed EARD framework, which is controlled by two key parameters, α and β. These parameters determine the mapping range of entity-level reliability weights and therefore play an important role in balancing the influence of user and item reliability during training. The optimal values were selected through a systematic grid search to ensure stable training behavior and strong empirical performance.

To account for variations in data characteristics, such as sparsity levels and noise distributions, as well as differences in model capacity, hyperparameters were tuned independently for each model and dataset combination. All reported results are based on test set performance under the selected configurations. The final optimal

Table 6: Optimal $[\alpha, \beta]$ values for EARD across backbone models and datasets, selected via grid search.

Model	ML-1M	Yelp	Amazon-book
GMF	[1.0, 2.0]	[0.9, 1.0]	[0.14, 0.4]
NeuMF	[0.5, 1.5]	[0.05, 0.1]	[0.05, 0.1]
CDAE	[0.5, 1.5]	[0.1, 0.5]	[0.1, 0.5]

$[\alpha, \beta]$ values are summarized in Table 6, facilitating reproducibility and fair comparison in future studies.

B Comparison of Hyperparameter Design

Beyond accuracy, practical usability is critical for deploying denoising frameworks in real-world systems. A key factor is the number and sensitivity of hyperparameters. Many state-of-the-art methods, though theoretically sound, introduce numerous non-intuitive hyperparameters that require extensive tuning, increasing computational cost and complicating deployment.

Table 7: Hyperparameter complexity of representative denoising methods. EARD uses only two parameters, reducing tuning overhead compared to DeCA, UDT, and SGDL.

Method	#Params	Hyperparameters and Brief Descriptions
DeCA	3	α: KL-divergence balance weight. C_+: Penalize negatives in positive-denoising. C_-: Penalize positives in negative-denoising.
UDT	4	a, b: Weight mapping range. a', b': Temperature mapping range.
SGDL	5	h: Memorization window size. η_2: Weight function learning rate. d_w: Weight function MLP size. d_l: Scheduler LSTM size. τ: Gumbel-Softmax temperature.
Ours	**2**	α, β: Entity weight mapping range.

To highlight EARD's practical advantages, Table 7 compares its hyperparameter design with strong baselines. Methods like DeCA, UDT, and SGDL rely on complex components (e.g., KL-divergence balancing, temperature scheduling, and meta-learning), each requiring careful tuning. In contrast, EARD uses only two intuitive hyperparameters, α and β, with stable, predictable effects. Empirical results show a smooth concave performance landscape (Figure 4; Appendix Figures 9 and 10), enabling efficient tuning and avoiding poor local optima. This simplicity and tunability reduce engineering overhead and enhance robustness in practice.

C Noise is Not Random: An Empirical Study of Entity-Level Bias

We find that noise in implicit feedback is systematic and closely linked to users and items. Across ML-1M, Yelp, and Amazon-Book, entity-level noise shows broad, non-uniform distributions. Defining noise rate as the proportion of interactions with ratings ≤ 3, we observe that both users and items span the full 0%–100% noise range in all datasets (Figures 6 and 1a).

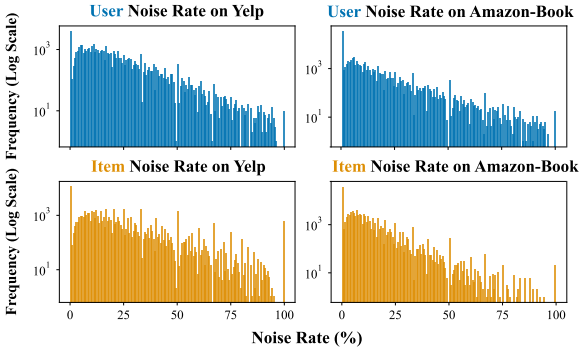

Figure 6: Histograms of user (blue) and item (yellow) noise rates on the ML-1M, Yelp, and Amazon-Book datasets.

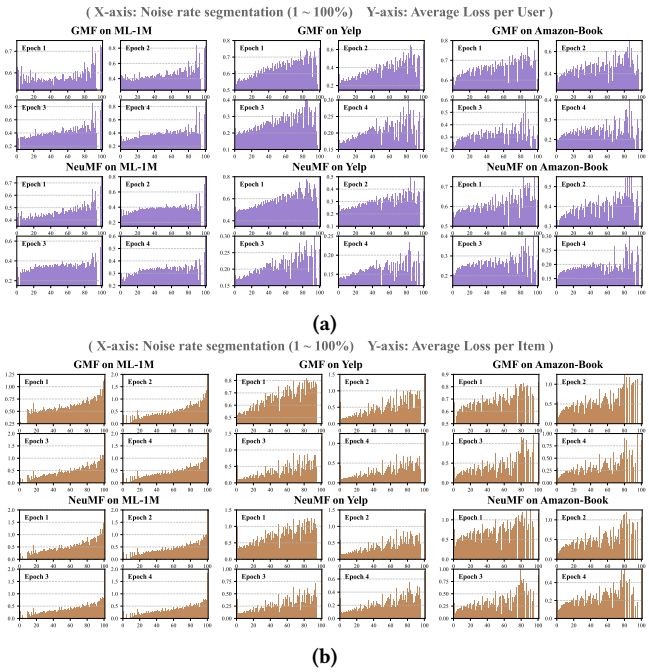

Figure 7: Loss patterns by entity noise rate. (a) Mean per-user and (b) mean per-item loss over 100 noise-rate percentiles. Higher noise rates consistently correspond to higher training loss, supporting their use as reliability proxies.

Crucially, a non-negligible number of entities exists within each segment of the noise distribution. This indicates that both highly reliable (low-noise) and highly unreliable (high-noise) entities are consistently present, regardless of the dataset's specific characteristics. This widespread heterogeneity invalidates the assumption that all entities are equally trustworthy and provides a strong motivation for our proposed EARD framework, which adaptively estimates user reliability and item reputation for more robust learning.

D Analyzing the Correlation Between Loss Patterns and Noise Rates

Figure 7 examines noise-aligned behavior from user and item perspectives. Training instances are grouped by interaction noise rate (1%–100%), and percentile-wise average losses are computed over the first four epochs. Figure 7a presents per-user loss distributions for GMF and NeuMF on ML-1M, Yelp, and Amazon-Book. In all settings, average user loss increases monotonically with noise. For example, under GMF on Yelp at Epoch 4, the top 20% noise group shows losses of 0.6–0.8, versus 0.2–0.3 for the bottom 20%, indicating that user-level loss reflects reliability. Figure 7b shows consistent trends at the item level, where higher-noise items incur larger losses, further supporting loss as a proxy for noisy interactions.

These results empirically support the core intuition behind EARD: entity-level loss serves as a reliable and quantifiable signal of interaction trustworthiness. The clear alignment between loss and noise across users and items validates loss-based metrics for adaptive reweighting and denoising.

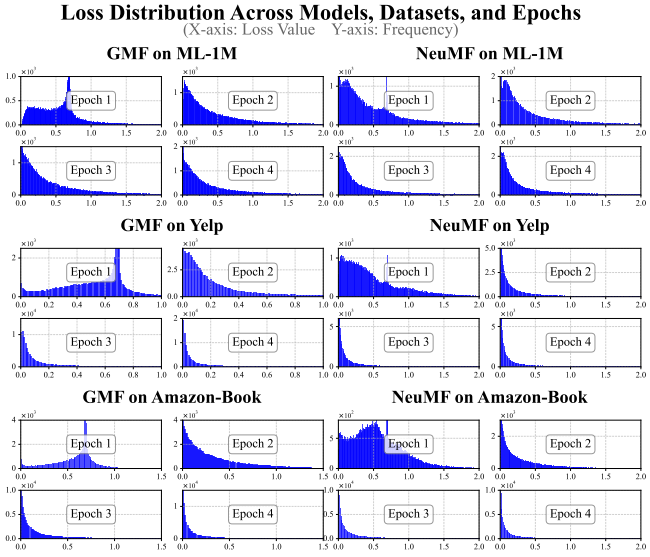

Figure 8: Histograms of training loss distributions over the first four epochs for GMF and NeuMF on the ML-1M, Yelp, and Amazon-Book datasets.

E Analysis of Loss Distribution Diversity Across Models and Datasets

Figure 8 shows instance-level loss distributions over the first four epochs for GMF and NeuMF on ML-1M, Yelp, and Amazon-Book. Each subplot presents a loss histogram for a given epoch. Loss distributions vary substantially across models, datasets, and epochs, making fixed distributional assumptions unrealistic. In practice, losses are often non-standard, long-tailed, and skewed with many outliers, which limits parametric modeling. Consequently, weighting strategies should be distribution-agnostic and robust. Parametric methods (e.g., GMM) and heuristics such as Top-k rely on assumptions (e.g., Gaussianity or unimodality) that are violated, whereas rank-based methods like ECDF are assumption-free and robust, motivating our use of ECDF for noise-aware reweighting.

F Base Weighting Strategies Analysis

F.1 ECDF-based Weighting

Algorithm 1 ECDF-based Weighting (Our Method)

Require:
Input: Set of training instances $\mathcal{D}^{(t)} = \{(u, i, y_{ui})\}$ for epoch t.
Input: Corresponding loss values $L = \{\ell_{ui} \mid (u, i, y_{ui}) \in \mathcal{D}^{(t)}\}$.
Ensure: **Output:** Confidence weights $W = \{w_{ui} \mid (u, i, y_{ui}) \in \mathcal{D}^{(t)}\}$ for each instance.
$n \leftarrow |L|$ {Total number of training instances}
$\boldsymbol{\ell}_{\text{valid}} \leftarrow \{-\ell_{ui} \mid \ell_{ui} \in L\}$
$\boldsymbol{\ell}_{\text{sorted}} \leftarrow \text{Sort}(\boldsymbol{\ell}_{\text{valid}})$
$\mathbf{r} \leftarrow \text{SearchSorted}(\boldsymbol{\ell}_{\text{sorted}}, \boldsymbol{\ell}_{\text{valid}})$ {Get rank (starting from 1) for each value in $\boldsymbol{\ell}_{\text{valid}}$ within the sorted list}
For each instance (u, i) **with loss** ℓ_{ui}:
  $w_{ui,\text{base}} \leftarrow (r_{ui} - 0.5)/n$ {using Hazen plotting position}
return $W \leftarrow \{w_{ui,\text{base}} \mid (u, i, y_{ui}) \in \mathcal{D}^{(t)}\}$

F.2 Gaussian Mixture Model (GMM)

Algorithm 2 Gaussian Mixture Model (GMM) Weighting

Require:
Input: Set of training instances $\mathcal{D}^{(t)} = \{(u, i, y_{ui})\}$ for epoch t.
Input: Corresponding loss values $L = \{\ell_{ui} \mid (u, i, y_{ui}) \in \mathcal{D}^{(t)}\}$.
Ensure: **Output:** Confidence weights $W = \{w_{ui} \mid (u, i, y_{ui}) \in \mathcal{D}^{(t)}\}$.
$n \leftarrow L$ {Total number of training instances in the current epoch}
$\boldsymbol{\ell}_{\text{neg}} \leftarrow \{-\ell_{ui} \mid \ell_{ui} \in L\}$ {Critical: Negate losses. Higher value in ℓ_{neg} indicates cleaner sample}
Fit a 2-component Gaussian Mixture Model (GMM) on the set $\boldsymbol{\ell}_{\text{neg}}$.
Let the two fitted Gaussian components be $k = 1, 2$ with means μ_1, μ_2.
$k_{\text{clean}} \leftarrow \arg\max_{k \in \{1,2\}}(\mu_k)$ {Identify the component with the larger mean as the 'clean' component}
For each value ℓ^{ui}_{neg} in $\boldsymbol{\ell}_{\text{neg}}$ {Iterate over the set of negated losses}
  $p_{ui} \leftarrow \text{PosteriorProbability}(\ell^{ui}_{\text{neg}} \text{ belongs to component } k_{\text{clean}})$
  $w_{ui} \leftarrow p_{ui}$
return $W \leftarrow \{w_{ui} \mid (u, i, y_{ui}) \in \mathcal{D}^{(t)}\}$

F.3 Hard Thresholding (Top-k Selection)

Hard Thresholding (Top-k Selection) is a denoising method based on the small-loss assumption, where lower-loss samples are more likely clean. At each epoch, instances are ranked by loss, and the top ρ fraction (e.g., $\rho = 0.8$) receive weight 1, while the rest are assigned 0, resulting in a binary selection scheme.

F.4 Linear Scaling

Linear Scaling assigns continuous weights by linearly mapping losses to $[0, 1]$, with the minimum loss mapped to 1 and the maximum to 0. While smooth, it is sensitive to outliers, which can distort scaling and suppress informative samples.

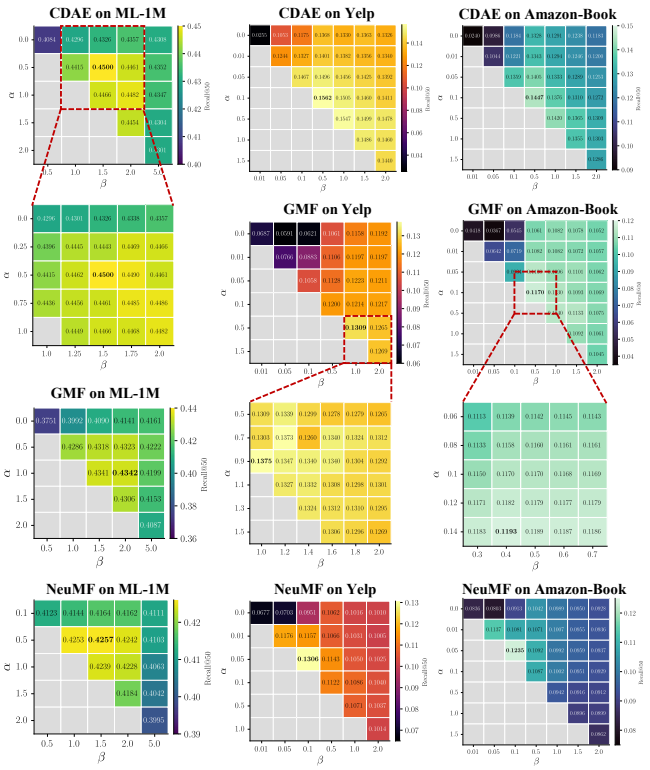

Figure 9: Hyperparameter sensitivity across models and datasets. Recall@50 heatmaps over α and β show smooth, near-concave behavior, indicating robustness.

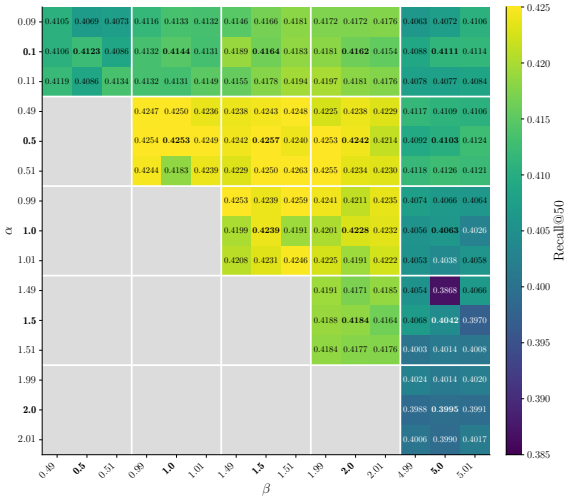

Figure 10: Heatmap of NeuMF performance on ML-1M (Recall@50), generated by finely sampling $[\alpha, \beta]$ around 15 anchor points (bolded). Brighter regions indicate better performance and align with Hessian-based analysis, empirically supporting local concavity in high-performance areas.

G Analysis of Hyperparameter Sensitivity

We analyze the impact of the weighting hyperparameters α and β on model performance and examine the geometry of the objective surface induced by EARD. Our study spans multiple models and datasets to evaluate both local and global landscape properties.

G.1 Generalization Across Models and Datasets

To assess whether approximate concavity generalizes across models and datasets, we plot Recall@50 heatmaps for GMF, NeuMF, and CDAE on ML-1M, Yelp, and Amazon-Book (Figure 9). The heatmaps, computed over (α, β) grids, consistently exhibit hill-like patterns, indicating global concavity across both simple and expressive models and datasets with varying sparsity and noise. These results suggest that the EARD objective is approximately concave in practice, with a clear local optimum, supporting stable optimization and robust hyperparameter design.

G.2 Concavity Verification via Hessian Matrix

We use a second-order approximation to analyze local curvature around selected (α, β) configurations. A discrete Hessian is computed from Recall@50 scores on a 3×3 grid at each anchor point. As outlined in Algorithm 3, second-order derivatives are estimated via finite differences, and curvature is evaluated using the Hessian determinant $\det(H)$. A point is considered locally concave if $H_{11} < 0$ and $\det(H) > 0$, indicating a hill-shaped landscape.

Algorithm 3 Verify Local Concavity via Discrete Hessian Test

Require: A 3×3 grid of function values $\mathbf{F}$ centered at $f_c = f(x, y)$, with step sizes h_x, h_y.
Ensure: Local curvature: Concave, Convex, Saddle Point, or Indeterminate.

{Compute second-order partial derivatives using finite differences from grid $\mathbf{F}$}
$H_{11} \leftarrow (F_{x+h,y} - 2f_c + F_{x-h,y})/h_x^2$
$H_{22} \leftarrow (F_{x,y+h} - 2f_c + F_{x,y-h})/h_y^2$
$H_{12} \leftarrow (F_{x+h,y+h} - F_{x-h,y+h} - F_{x+h,y-h} + F_{x-h,y-h})/(4h_xh_y)$
Compute determinant: $\det(H) \leftarrow H_{11}H_{22} - H_{12}^2$.
if $H_{11} < 0$ and $\det(H) > 0$ **then**
  return Locally Concave ('hill')
else if $H_{11} > 0$ and $\det(H) > 0$ **then**
  return Locally Convex ('basin')
else if $\det(H) < 0$ **then**
  return Saddle Point
else
  return Indeterminate

To assess surface smoothness and concavity, we conduct a fine-grained (α, β) sweep for NeuMF on ML-1M (Figure 10). Sampling about 15 anchor points at 0.01 resolution reveals a clear unimodal landscape, with peak performance at the center and smooth decay toward the edges. This agrees with the Hessian-based analysis in Figure 4b, where most points show local concavity. Overall, the results confirm a unimodal objective surface and the stability of EARD under small hyperparameter perturbations.

Definition: *$f(x, y)$ denotes the model's evaluation score (e.g., Recall@50) on the test set when trained with $\alpha = x$ and $\beta = y$.*