

# Denoising Diffusion Recommender Model

Jujia Zhao  
zhao.jujia.0913@gmail.com  
Leiden University  
Leiden, The Netherlands

Wenjie Wang\*  
wenjiewang96@gmail.com  
National University of Singapore  
Singapore

Yiyan Xu  
yiyanxu24@gmail.com  
University of Science and Technology  
of China  
Hefei, China

Teng Sun  
stbestforever@gmail.com  
Shandong University  
Qingdao, China

Fuli Feng  
fulifeng93@gmail.com  
University of Science and Technology  
of China  
Hefei, China

Tat-Seng Chua  
dcscts@nus.edu.sg  
National University of Singapore  
Singapore

## ABSTRACT

Recommender systems often grapple with noisy implicit feedback. Most studies alleviate the noise issues from data cleaning perspective such as data resampling and reweighting, but they are constrained by heuristic assumptions. Another denoising avenue is from model perspective, which proactively injects noises into user-item interactions and enhances the intrinsic denoising ability of models. However, this kind of denoising process poses significant challenges to the recommender model's representation capacity to capture noise patterns.

To address this issue, we propose Denoising Diffusion Recommender Model (DDRM), which leverages multi-step denoising process of diffusion models to robustify user and item embeddings from any recommender models. DDRM injects controlled Gaussian noises in the forward process and iteratively removes noises in the reverse denoising process, thereby improving embedding robustness against noisy feedback. To achieve this target, the key lies in offering appropriate guidance to steer the reverse denoising process and providing a proper starting point to start the forward-reverse process during inference. In particular, we propose a dedicated denoising module that encodes collaborative information as denoising guidance. Besides, in the inference stage, DDRM utilizes the average embeddings of users' historically liked items as the starting point rather than using pure noise since pure noise lacks personalization, which increases the difficulty of the denoising process. Extensive experiments on three datasets with three representative backend recommender models demonstrate the effectiveness of DDRM.

\*Corresponding author. This work is supported by the National Key Research and Development Program of China (2022YFB3104701) and the National Natural Science Foundation of China (62272437).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).  
SIGIR '24, July 14–18, 2024, Washington, DC, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 979-8-4007-0431-4/24/07  
<https://doi.org/10.1145/3626772.3657825>

## CCS CONCEPTS

• Information systems → Recommender systems.

## KEYWORDS

Denoising Recommendation, Diffusion Model, Noisy Implicit Feedback

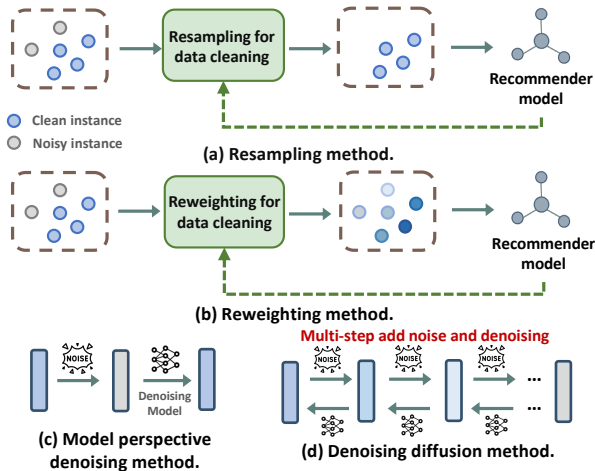
## ACM Reference Format:

Jujia Zhao, Wenjie Wang, 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 (SIGIR '24)*, July 14–18, 2024, Washington, DC, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3626772.3657825>

## 1 INTRODUCTION

Recommender systems play a pivotal role in personalized information delivery across a wide range of Web applications [12, 52, 56]. Typically, recommender models learn personalized user preferences from user feedback [29, 53]. Due to the ease of collecting implicit feedback (e.g., click and purchase) in large volume, it has become indispensable for user preference learning [33, 49, 54]. Unfortunately, implicit feedback inevitably contains noises [2, 24, 44]. For instance, clicks on micro-videos may not indicate users' actual satisfaction due to various interference factors [38]. Such noisy feedback misguides recommender models in interpreting user preferences, subsequently hampering the recommendation performance [8, 24, 43]. As such, denoising implicit feedback for recommendation becomes an imperative task [1].

Previous work primarily mitigates the impact of noisy feedback from the perspective of data cleaning, including resampling and reweighting user-item interactions. Specifically, 1) resampling methods [5, 6, 51] aim to identify noisy interactions and sample more clean interactions for training (Figure 1(a)). For instance, WBPR [9] believes that non-interacted popular items are more likely to be true negative items and allocates higher sampling probabilities. 2) Reweighting methods [38, 41, 43] utilize all training interactions yet assign lower weights to potential noisy ones (Figure 1(b)). For instance, reweighted loss [38] assigns lower weights to the large-loss interactions since it assumes large-loss interactions are more likely to be noisy. Notably, these data cleaning methods depend on certain heuristic assumptions, such as the large-loss assumption [38] and cross-model agreement [41]. As their



**Figure 1: Illustration of resampling, reweighting, model perspective denoising, and denoising diffusion methods.**

assumptions rely heavily on the distribution of noisy interactions, these data cleaning methods suffer from limited adaptability, requiring substantial configuration tuning to adapt to different backend models and datasets.

For denoising implicit feedback, another research line is from model perspective, seeking to enhance recommender models’ inherent noise resistance capabilities. These model perspective methods usually add random noises to user-item interactions [48] or drop positive interactions as augmented data [11, 45], and then regulate recommender models to learn robust representations from the augmented data [42]. For example, CDAE corrupts users’ interactions randomly with a noise ratio and subsequently optimizes recommender models to recover the original clean interactions. Nevertheless, as illustrated in Figure 1(c), these model perspective methods solely rely on a denoising model to directly convert noisy data into clean data, imposing substantial demands on the model’s representation capacity to efficiently capture noise patterns.

Diffusion models, as a kind of powerful generative model, inherently possess a denoising aptitude to enhance existing model perspective denoising methods [14]. Diffusion models have already revealed remarkable representation capabilities across various domains like image generation and molecule generation [4, 35]. The potential benefits lie in two aspects: 1) during the forward process, diffusion models enhance noise diversity by injecting noises with controllable noise scales and steps, potentially leading to the capability to denoise a broader spectrum of noise. and 2) in the reverse denoising process, diffusion models decompose the complex denoising problem into multiple steps, thereby reducing the denoising difficulty at each step (Figure 1(d)). In light of these, it is promising to leverage diffusion models to robustify user and item representations from existing recommender models.

To this end, we propose a plug-in denoising model for existing recommender models called **Denoising Diffusion Recommender Model (DDRM)**. In the training stage, given user and item embeddings from any recommender models, DDRM improves their robustness against noisy feedback via a forward-reverse process. In the forward process, we proactively inject Gaussian noises into user and item embeddings with adjustable scales and steps, yielding

noisy embeddings. The reverse denoising process then iteratively removes noises via a learnable neural network. During the inference phase, given a user, DDRM should choose a starting point to initialize this refined forward-reverse process, thereby generating an ideal item that aligns closely with the user’s preferences.

To achieve the denoising target with DDRM, the key lies in 1) offering appropriate guidance to steer the reverse denoising process and 2) providing a proper starting point to start the forward-reverse process during inference. As for the guidance in the denoising process, our goal is to derive effective representations from weak collaborative information, which helps to guide the denoising process towards the clean embeddings of users (or items). As for the starting point, we aim to incorporate personalized information rather than relying solely on pure noise. This strategy diverges from traditional diffusion models, which typically generate images from pure noise guided by abundant textual instructions [21]. In the recommendation scenario, guidance through collaborative information is not as explicit as textual instructions, making it challenging to generate personalized ideal items if we use pure noise as the starting point (see evidence in Table 3). By incorporating personalized information, it’s more likely to generate the ideal item closely aligned with individual user preference.

Toward these goals, we design a denoising module for the reverse process of DDRM. Given noisy user (or item) embeddings, the denoising module devises strategies to encode the collaborative information, *e.g.*, users’ liked items, to guide the reverse denoising process. For the inference phase, to generate an ideal item as the recommendation for a user, we take the average embeddings of the user’s historically liked items as the starting point, instead of using pure noise (*cf.* Section 3.3 for details). Given the embedding of the generated item, we present a rounding function to ground the generated item to existing item candidates by the embeddings’ similarity. As an extension, we also consider adding a reweighted loss to supplement DDRM from the perspective of data cleaning. We implement DDRM on three representative recommender models and conduct comprehensive experiments to validate its effectiveness against other baselines on three public datasets.

The main contributions of this work are threefold:

- We propose a model-agnostic denoising diffusion recommender model, which robustifies the user and item representations from any recommender models against noisy feedback.
- We design the user and item denoising modules to denoise the user and item embeddings, in which we incorporate collaborative information as guidance to steer the denoising process, and integrate personalized information as the starting point during inference.
- We instantiate DDRM on three backend models and execute extensive experiments under various settings, confirming its efficacy across three public datasets<sup>1</sup>.

## 2 PRELIMINARY

Diffusion models have already demonstrated proficient performance in domains like computer vision and molecular generation [16, 31]. Typically, diffusion models encompass two components: the forward and reverse processes [14].

<sup>1</sup>Our code and data are released at <https://github.com/Polaris-JZ/DDRM>.

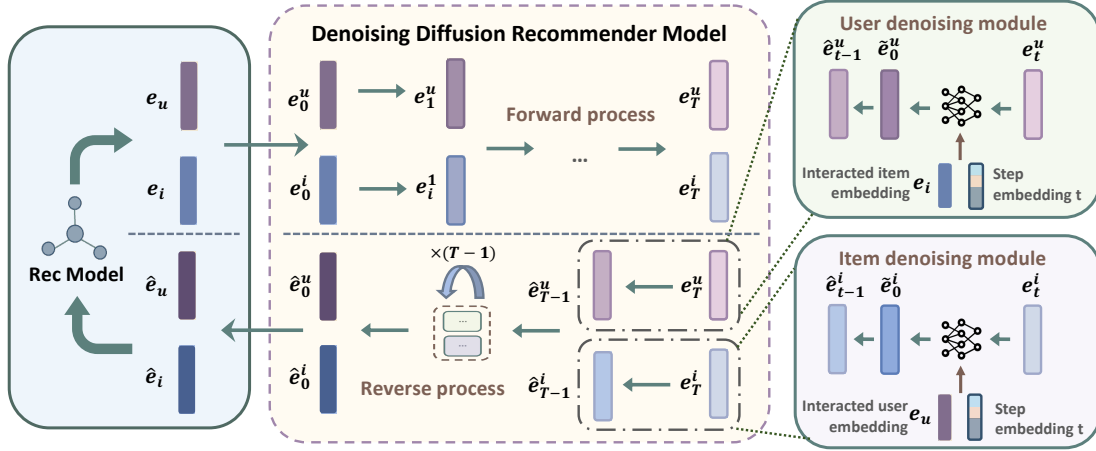


Figure 2: Structure of DDRM. The left part is the backend recommender model. DDRM accepts both user and item embeddings as inputs and subsequently produces denoised embeddings that are fed back into the model to do the recommendation task.

• **Forward process** aims to inject Gaussian noises into the original data. Given a data sample  $x_0$ , diffusion models continuously add different scale Gaussian noises to it in  $T$  steps until get  $x_T$ . Specifically, for adding noise from  $x_{t-1}$  to  $x_t$ , we have:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t, \sqrt{1 - \beta_t}x_{t-1}, \beta_t\mathbf{I}), \quad (1)$$

where  $t \in \{1, 2, \dots, T\}$  is the current step,  $\beta_t \in (0, 1)$  is the noise scale in step  $t$ ,  $\mathbf{I}$  is the identity matrix, and  $\mathcal{N}$  is the Gaussian distribution which means  $x_t$  is sampled from this distribution. According to the additivity of independent Gaussian noises and reparameterization trick [14, 15],  $x_t$  can be directly obtained from  $x_0$  in the calculation:

$$q(x_t|x_0) = \mathcal{N}(x_t, \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}), \quad (2)$$

where  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ ,  $\alpha_s = 1 - \beta_s$ .

• **Reverse process** is designed to iteratively denoise the noisy data  $x_T$ , following the sequence  $(x_T \rightarrow x_{T-1} \rightarrow x_{T-2} \rightarrow \dots \rightarrow x_0)$ . According to [50], under the conditions where  $q(x_t|x_{t-1})$  conforms to a Gaussian distribution and  $\beta_t$  remains sufficiently small, the distribution  $p(x_{t-1}|x_t)$  also exhibits Gaussian properties. As such, a neural network can be utilized to predict this reverse distribution:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \quad (3)$$

where  $\theta$  is the parameters of the neural network, and  $\mu_\theta(x_t, t)$  and  $\Sigma_\theta(x_t, t)$  are the mean and covariance of this Gaussian distribution.

• **Training.** For training the diffusion models, the key focus is obtaining reliable values for  $\mu_\theta(x_t, t)$  and  $\Sigma_\theta(x_t, t)$  to guide the reverse process towards accurate denoising. To achieve this, it is important to optimize the variational lower bound of the negative log-likelihood of the model's predictive denoising distribution  $p_\theta(x_0)$ :

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_{q(x_0)} [-\log p_\theta(x_0)] \\ &\leq \mathbb{E}_q [L_T + L_{T-1} + \dots + L_0], \text{ where} \end{aligned} \quad (4)$$

$$\begin{cases} L_T = D_{\text{KL}}(q(x_T|x_0) \parallel p_\theta(x_T)) \\ L_t = D_{\text{KL}}(q(x_t|x_{t+1}, x_0) \parallel p_\theta(x_t|x_{t+1})) \\ L_0 = -\log p_\theta(x_0|x_1), \end{cases} \quad (5)$$

where  $t \in \{1, 2, \dots, T-1\}$ . While  $L_T$  can be disregarded during training due to the absence of learnable parameters in the forward process,  $L_0$  represents the negative log probability of the original

data sample  $x_0$  given the first-step noisy data  $x_1$ , and  $L_t$  aims to align the distribution  $p_\theta(x_t|x_{t+1})$  with the tractable posterior distribution  $q(x_t|x_{t+1}, x_0)$  in the reverse process [26].

### 3 METHOD

To mitigate the effect of noisy implicit feedback, we propose DDRM to denoise the user and item embeddings. As illustrated in Figure 2, given pre-trained user and item embeddings, DDRM continuously injects Gaussian noises and then denoises these noisy embeddings iteratively through a forward-reverse process. For effective guidance during denoising, DDRM incorporates specialized user and item denoising modules within the reverse process. During the inference phase, DDRM applies this refined forward-reverse process to a specifically chosen starting point, thereby generating an ideal item that aligns closely with user's preferences.

#### 3.1 DDRM Framework

• **Forward process.** Given pre-trained user embeddings  $e_u$  of user  $u$  and item embeddings  $e_i$  of item  $i$  from a backend recommender model, we begin the forward process by setting  $e_0^u = e_u$  and  $e_0^i = e_i$ . Subsequently, we continuously incorporate Gaussian noises into  $e_0^u$  and  $e_0^i$  separately with adjustable scales and steps:

$$q(e_t^u|e_0^u) = \mathcal{N}(e_t^u, \sqrt{\bar{\alpha}_t}e_0^u, (1 - \bar{\alpha}_t)\mathbf{I}), \quad (6)$$

$$q(e_t^i|e_0^i) = \mathcal{N}(e_t^i, \sqrt{\bar{\alpha}_t}e_0^i, (1 - \bar{\alpha}_t)\mathbf{I}), \quad (7)$$

where  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$ ,  $\alpha_s = 1 - \beta_s$ ,  $\beta_s \in (0, 1)$  controls the noise scale added to the embedding in the current step  $s$ , and  $e_t^{(\cdot)}$  denotes the user or item embeddings in the forward step  $t$ . To regulate the noise level in each step, we follow [39] employing a *linear variance* noise schedule in the forward process:

$$1 - \bar{\alpha}_t = s \cdot \left[ \alpha_{\min} + \frac{t-1}{T-1} (\alpha_{\max} - \alpha_{\min}) \right], \quad (8)$$

where  $\alpha_{\min}$  and  $\alpha_{\max}$  are the minimum and maximum of the noise correspondingly,  $t$  is the current forward step,  $T$  is the total forward step, and  $s \in (0, 1)$  controls the noise scale.

• **Reverse process.** After getting noisy user embeddings  $e_t^u$  and noisy item embeddings  $e_t^i$  in the forward process, we denoise these embeddings iteratively in the reverse process. In each reverse step, we design the user denoising module and the item denoising module to denoise user embeddings and item embeddings separately since the noise distribution is different for users and items:

$$p_\theta(\hat{e}_{t-1}^u | \hat{e}_t^u) = \mathcal{N}(\hat{e}_{t-1}^u; \mu_\theta(\hat{e}_t^u, t), \Sigma_\theta(\hat{e}_t^u, t)), \quad (9)$$

$$p_\psi(\hat{e}_{t-1}^i | \hat{e}_t^i) = \mathcal{N}(\hat{e}_{t-1}^i; \mu_\psi(\hat{e}_t^i, t), \Sigma_\psi(\hat{e}_t^i, t)), \quad (10)$$

where  $\hat{e}_t^u$  and  $\hat{e}_t^i$  are the denoised embeddings in the reverse step  $t$ ,  $\theta$  and  $\psi$  are the learnable parameters of the user denoising module and the item denoising module correspondingly. These denoising module are executed iteratively in the reverse process until the generation of final clean embeddings  $\hat{e}_0^u$  and  $\hat{e}_0^i$ .

• **Denoising module.** Denoising module aims to denoise the noisy embedding in each reverse step. Since the user denoising module and the item denoising module have the same structure, we mainly focus on explaining the user denoising module as formulated as Eq. (9). As illustrated in Eq. (5), the diffusion training aims to align the distribution  $p_\theta(\hat{e}_{t-1}^u | \hat{e}_t^u)$  with the tractable posterior distribution  $q(e_{t-1}^u | e_t^u, e_0^u)$  in the reverse process, thus we can use  $q(e_{t-1}^u | e_t^u, e_0^u)$  to constrain  $p_\theta(\hat{e}_{t-1}^u | \hat{e}_t^u)$ . Through Bayes' theorem, we can derive:

$$q(e_{t-1}^u | e_t^u, e_0^u) = \mathcal{N}(e_{t-1}^u; \tilde{\mu}_t(e_t^u, e_0^u), \tilde{\beta}_t I), \quad \text{where} \quad (11)$$

$$\begin{cases} \tilde{\mu}_t(e_t^u, e_0^u) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} e_t^u + \frac{\sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)}{1 - \bar{\alpha}_t} e_0^u, \\ \tilde{\beta}_t = \frac{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}. \end{cases} \quad (12)$$

$\tilde{\mu}_t(e_t^u, e_0^u)$  and  $\tilde{\beta}_t I$  are the mean and covariance of  $q(e_{t-1}^u | e_t^u, e_0^u)$ . Following [39], we can similarity factorize  $p_\theta(\hat{e}_{t-1}^u | \hat{e}_t^u)$ :

$$p_\theta(\hat{e}_{t-1}^u | \hat{e}_t^u) = \mathcal{N}(\hat{e}_{t-1}^u; \mu_\theta(\hat{e}_t^u, t), \tilde{\beta}_t I), \quad \text{where} \quad (13)$$

$$\mu_\theta(\hat{e}_t^u, t) = \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \hat{e}_t^u + \frac{\sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)}{1 - \bar{\alpha}_t} \tilde{e}_0^u. \quad (14)$$

$\tilde{e}_0^u$  is the predicted  $e_0^u$  since the distribution of  $e_0^u$  is unknown in the reverse process. We employ the multi-layer perceptron (MLP) to reconstruct  $\tilde{e}_0^u$  in the denoising module.

To ensure tractability in the embedding denoising process, it is paramount to utilize specific conditions as guidance. In this context, DDRM incorporates two key conditions during the denoising phase: collaborative information and step information. Collaborative information, which is based on user interaction behaviors, can enable the denoising module to recognize user preferences, subsequently assisting to identify and mitigate noise. Additionally, the step information also affects the denoising performance. It provides insight into the current noise level in the embeddings, offering a gauge on the extent of denoising required at each step. As such, we use collaborative information  $\mathbf{c}_u$  and step embedding as condition elements, to guide the reconstruction process in the denoising modules (*cf.* Section 3.2 for the calculation of  $\mathbf{c}_u$  in detail). Specifically, for user embedding  $\hat{e}_t^u$  in the reverse step  $t$ , the user reconstruction MLP yields:

$$\tilde{e}_0^u = f_\theta(\hat{e}_t^u, \mathbf{c}_u, t), \quad (15)$$

where  $\tilde{e}_0^u$  is the predicted  $e_0^u$  by the reconstruction MLP with the parameter  $\theta$ . It is noteworthy that the step information is encoded

### Algorithm 1 DDRM Training

**Input:** interaction data  $\bar{D}$ , pre-trained user embedding  $e_0^u$ , pre-trained item embedding  $e_0^i$ , diffusion step  $T$ , user reconstruction MLP  $f_\theta$ , item reconstruction MLP  $f_\psi$

```

1: repeat
2:   Sample a batch of interactions  $D \subset \bar{D}$ .
3:   for all  $(u, i, j) \in D$  do
4:     Sample  $t \sim \mathcal{U}(1, T)$ 
5:     Compute  $e_t^u$  given  $e_0^u$  and  $t$  via  $q(e_t^u | e_0^u)$  in Eq. (6);
6:     Compute  $e_t^i$  given  $e_0^i$  and  $t$  via  $q(e_t^i | e_0^i)$  in Eq. (7);
7:     Reconstruct  $\tilde{e}_0^u$  and  $\tilde{e}_0^i$  through  $f_\theta$  and  $f_\psi$ ;
8:     Calculate  $\mathcal{L}_{\text{final}}$  by Eq. (22);
9:     Take gradient descent step on  $\nabla_\theta(\mathcal{L}_{\text{final}})$  to optimize  $\theta$ ;
10:    Take gradient descent step on  $\nabla_\psi(\mathcal{L}_{\text{final}})$  to optimize  $\psi$ ;
11: until converged
Output: optimized  $\theta, \psi$ .

```

through sinusoidal positional encoding [14], and these three inputs are concatenated together to feed into the MLP.

Similarly, for the item denoising module, given item embedding  $\hat{e}_t^i$  in the reverse step  $t$ , the other item reconstruction MLP outputs  $\tilde{e}_0^i = f_\psi(\hat{e}_t^i, \mathbf{c}_i, t)$ , where  $\mathbf{c}_i$  represent the collaborative information of item  $i$ , and  $\tilde{e}_0^i$  is the predicted  $e_0^i$  by the item reconstruction MLP with the parameter  $\psi$ .

Generally, in the denoising module, we get predicted original embedding  $\tilde{e}_0^u$  and  $\tilde{e}_0^i$  from the user reconstruction MLP and the item reconstruction MLP, respectively, and then utilize Eq. (13) to get the denoised embeddings for the current step.

## 3.2 Optimization

• **DDRM training.** The optimization of DDRM is under the BPR training setting: given recommendation data with the triplet  $(u, i, j)$ , item  $i$  and item  $j$  are the positive item and negative item of user  $u$ , respectively. Please note that we only conduct denoising for user  $u$  and positive interacted item  $i$  since denoising negative item  $j$  is rendered relatively insignificant due to the random negative sampling mechanism.

To optimize the embedding denoising process, it is essential to minimize the variational lower bound of the predicted user and item embeddings. According to the KL divergence based on the multivariate Gaussian distribution in Eq. (5), the reconstruction loss of the denoising process within a training iteration is expressed as:

$$\begin{aligned} \mathcal{L}_{\text{re}}(u, i) &= \mathbb{E}_q[-\log p_\theta(\hat{e}_0^u) - \log p_\psi(\hat{e}_0^i)] \\ &\leq \mathcal{L}^u + \mathcal{L}^i + \mathcal{L}_0^u + \mathcal{L}_0^i, \quad \text{where} \end{aligned} \quad (16)$$

$$\begin{cases} \mathcal{L}^u = \sum_{t=2}^T \mathbb{E}_q \left[ \frac{1}{2} \left( \frac{\bar{\alpha}_{t-1}}{1 - \bar{\alpha}_{t-1}} - \frac{\bar{\alpha}_t}{1 - \bar{\alpha}_t} \right) \| e_0^u - \hat{f}_\theta(e_t^u, \mathbf{e}_i, t) \|_2^2 \right], \\ \mathcal{L}^i = \sum_{t=2}^T \mathbb{E}_q \left[ \frac{1}{2} \left( \frac{\bar{\alpha}_{t-1}}{1 - \bar{\alpha}_{t-1}} - \frac{\bar{\alpha}_t}{1 - \bar{\alpha}_t} \right) \| e_0^i - \hat{f}_\psi(e_t^i, \mathbf{e}_u, t) \|_2^2 \right], \\ \mathcal{L}_0^u = \mathbb{E}_q \left[ \| e_0^u - \hat{f}_\theta(e_1^u, \mathbf{e}_i, 1) \|_2^2 \right], \\ \mathcal{L}_0^i = \mathbb{E}_q \left[ \| e_0^i - \hat{f}_\psi(e_1^i, \mathbf{e}_u, 1) \|_2^2 \right], \end{cases} \quad (17)$$

where  $\mathbf{e}_i$  and  $\mathbf{e}_u$  are the original embeddings of user  $u$  and item  $i$ , which serve as the collaborative information  $\mathbf{c}_u$  and  $\mathbf{c}_i$ , respectively.

**Algorithm 2** DDRM Inference

**Input:** all users  $\bar{U}$ , diffusion step  $T$ , item reconstruction MLP  $f_\psi$

- 1: Sample a batch of users  $U \subset \bar{U}$ .
- 2: **for all**  $u \in U$  **do**
- 3:   Compute average embeddings of users' historically liked items  $\bar{e}_i$  via Eq. (23);
- 4:   Compute  $\hat{e}_T^i$  given  $\bar{e}_0^i$  and  $T$  via  $q(\hat{e}_T^i | \bar{e}_0^i)$  in Eq. (7);
- 5:   **for**  $t = T, \dots, 1$  **do**
- 6:     Reconstruct  $\hat{e}_0^i$  through  $f_\psi$ ;
- 7:     Compute  $\hat{e}_{t-1}^i$  from  $\hat{e}_t^i$  and  $\bar{e}_0^i$  via Eq. (13);
- 8:   Rounding via Eq. (24) to get the ideal item embedding  $e$ ;

**Output:** the ideal item embedding  $e$ .

$\mathcal{L}^u$  and  $\mathcal{L}^i$  are the user and item reconstruction loss in the reverse process,  $\mathcal{L}_0^u$  and  $\mathcal{L}_0^i$  are the final prediction loss correspondingly. From Eq. (18), it is clear that the essence of the DDRM training lies in optimizing the distance between the reconstructed embedding derived from MLP and the original embedding.

To reduce the computational cost in the implementation, we simplify Eq. (17) by uniformly sampling  $t$  from  $\{1, 2, \dots, T\}$  instead of summing  $T$  steps and removing the weight before the MSE terms:

$$\mathcal{L}_{\text{re}}(u, i) = (\mathcal{L}_{\text{simple}}^u + \mathcal{L}_{\text{simple}}^i)/2, \quad \text{where} \quad (18)$$

$$\begin{cases} \mathcal{L}_{\text{simple}}^u = \mathbb{E}_{t \sim \mathcal{U}(1, T)} \mathbb{E}_q [\|\mathbf{e}_0^u - \hat{\mathbf{e}}_t^u(\mathbf{e}_t^u, \mathbf{e}_i, t)\|_2^2], \\ \mathcal{L}_{\text{simple}}^i = \mathbb{E}_{t \sim \mathcal{U}(1, T)} \mathbb{E}_q [\|\mathbf{e}_0^i - \hat{\mathbf{e}}_t^i(\mathbf{e}_t^i, \mathbf{e}_u, t)\|_2^2]. \end{cases} \quad (19)$$

• **Loss function.** The final loss function of DDRM comprises two parts: a reconstruction loss for the denoising process and a BPR loss for the recommendation task. The reconstruction loss  $\mathcal{L}_{\text{re}}(u, i)$  is derived from Eq. (18), which regulates the denoising of user and item embeddings.

After obtaining the denoised user and positive item embeddings via DDRM, these embeddings contribute to the computation of BPR loss  $\mathcal{L}_{\text{bpr}}$  [13]. We design a loss balance factor  $\lambda$  to adjust the weight of these two losses:

$$\mathcal{L}(u, i, j) = \lambda \mathcal{L}_{\text{bpr}}(u, i, j) + (1 - \lambda) \mathcal{L}_{\text{re}}(u, i), \quad (20)$$

where  $i$  and  $j$  are positive and negative items for user  $u$  in the BPR training setting. As an extension, we also consider adding a reweighted loss to supplement DDRM from the perspective of data cleaning (see empirical evidence of its effectiveness in Section 4.3.2). Specifically, inspired by [38], we dynamically allocate lower weights to instances with relatively lower positive scores since they are more likely to be noisy data.

$$w(u, i, j) = \text{sigmoid}(s(u, i))^\gamma, \quad (21)$$

$$\mathcal{L}_{\text{final}}(u, i, j) = w(u, i, j) \mathcal{L}(u, i, j), \quad (22)$$

where  $s(u, i)$  quantifies the score between users  $u$  and positive items  $i$ , and  $\gamma$  is the reweighted factor which controls the range of weights. The training step of DDRM is illustrated in Algorithm 1.

### 3.3 Inference

In the inference phase, we need to apply the refined forward-reverse process to a starting point, thereby generating ideal items to do the recommendation task for each user. Since guidance through collaborative information is not as explicit as textual instructions

**Table 1: Statistics of three datasets under two distinct settings. “#Int.” denotes interactions numbers. “N” and “R” represent natural noise setting and random noise setting, respectively.**

	#User	#Item (N)	#Int. (N)	#Item (R)	#Int. (R)
<b>Yelp</b>	54,574	77,405	1,471,675	34,395	1,402,736
<b>Amazon-book</b>	108,822	178,181	3,145,223	94,949	3,146,256
<b>ML-1M</b>	5,949	3,494	618,297	2,810	571,531

used in fields like image generation, we incorporate personalized information into the starting point rather than using pure noise. Particularly, we take the average embeddings of users' historically liked items as input since the interaction information reflects the preferences of users, thereby it's more likely to align the generated ideal item closely with individual user needs. Specifically, we first get the average item embedding  $\bar{e}_i$ :

$$\bar{e}_i = \frac{1}{n} \sum_{i \in \mathcal{I}_u} \mathbf{e}_i, \quad (23)$$

where  $n$  is the number of historical interacted items of user  $u$ , and  $\mathcal{I}_u$  is the historical interacted items. Subsequently, we introduce noise into  $\bar{e}_i$  continuously following the sequence  $\bar{e}_0^i \rightarrow \bar{e}_1^i \rightarrow \dots \rightarrow \bar{e}_T^i$  in the forward process. And then, we set  $\hat{e}_T^i = \bar{e}_T^i$  to execute the reverse process by  $\hat{e}_T^i \rightarrow \hat{e}_{T-1}^i \rightarrow \dots \rightarrow \hat{e}_0^i$  to generate a new item embedding  $\hat{e}_0^i$  conditioned on current step embedding and user original embedding  $\mathbf{e}_u$ . Following this, to obtain ideal items for the recommendation task, we develop a rounding function  $s(\hat{e}_0^i, \mathbf{e}_i)$  that calculates the inner product between the generated item embeddings  $\hat{e}_0^i$  and the candidate item embeddings to get a similarity score:

$$s(\hat{e}_0^i, \mathbf{e}_i) = \hat{e}_0^i \cdot \mathbf{e}_i, \quad i \in \mathcal{I} \quad (24)$$

where  $\mathcal{I}$  is the candidate item pool. Subsequently, we rank the similarity score and select the top- $k$  candidate items for recommendation. The inference procedure of DDRM is stated in Algorithm 2.

## 4 EXPERIMENTS

In this section, we conduct a comprehensive experimental study to address the following research questions:

- **RQ1:** How does the performance of DDRM compare with other baselines across the datasets in different experiment settings?
- **RQ2:** What is the impact of different components (e.g., reconstruction loss and reweighted loss, user and item denoising modules) within the DDRM on overall performance?
- **RQ3:** How do design variations in DDRM influence efficacy?

### 4.1 Experimental Settings

**4.1.1 Datasets.** We evaluate our proposed DDRM on three publicly accessible datasets in different experiment settings. 1) **Yelp**<sup>2</sup> contains a large collection of user reviews and ratings for different restaurants. 2) **Amazon-book**<sup>3</sup> covers users' purchase history and rating scores over books. 3) **ML-1M**<sup>4</sup> compiles movie ratings

<sup>2</sup><https://www.yelp.com/dataset/>.

<sup>3</sup><https://jmcauley.ucsd.edu/data/amazon/>.

<sup>4</sup><https://grouplens.org/datasets/movielens/1m/>.

**Table 2: Overall performance of DDRM and other baselines under natural noise setting. Bold signifies the best performance among the backend models, model-agnostic methods and DDRM. \* denotes statistically significant improvements of DDRM over the backend models, according to the t-tests with a significance level of  $p < 0.01$ .**

Methods	Yelp				Amazon-book				ML-1M			
	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20
AdaGCL	0.0464	0.0774	0.0277	0.0371	0.0282	0.0464	0.0166	0.0220	0.0630	0.1200	0.0453	0.0659
CDAE	0.0305	0.0530	0.0178	0.0246	0.0219	0.0399	0.0122	0.0175	0.0355	0.0675	0.0272	0.0391
MultiVAE	0.0484	0.0823	0.0289	0.0391	0.0508	0.0771	0.0300	0.0379	0.0636	0.1229	0.0450	0.0667
DiffRec	0.0501	0.0847	0.0307	0.0412	0.0537	0.0806	0.0329	0.0411	0.0658	0.1236	0.0488	0.0703
MFBPR	0.0286	0.0503	0.0176	0.0242	0.0217	0.0379	0.0131	0.0179	0.0445	0.0890	0.0429	0.0582
+T-CE	0.0316	0.0547	0.0191	0.0261	0.0230	0.0393	0.0136	0.0185	0.0460	0.0866	0.0432	0.0573
+R-CE	0.0324	0.0554	0.0195	0.0265	0.0227	0.0394	0.0134	0.0184	0.0472	0.0901	0.0434	0.0587
+DeCA	0.0295	0.0494	0.0181	0.0241	0.0118	0.0188	0.0071	0.0092	0.0451	0.0863	0.0428	0.0851
+BOD	0.0331	0.0568	0.0201	0.0269	0.0225	0.0376	0.0138	0.0185	0.0458	0.0892	0.0431	0.0585
+DDRM	<b>0.0358*</b>	<b>0.0582*</b>	<b>0.0217*</b>	<b>0.0285*</b>	<b>0.0249*</b>	<b>0.0406*</b>	<b>0.0148*</b>	<b>0.0196*</b>	<b>0.0477*</b>	<b>0.0916*</b>	<b>0.0450*</b>	<b>0.0601*</b>
LightGCN	0.0502	0.0858	0.0295	0.0403	0.0432	0.0710	0.0251	0.0333	0.0618	0.1193	0.0444	0.0652
+T-CE	0.0504	0.0856	0.0294	0.0400	0.0421	0.0691	0.0242	0.0323	0.0625	0.1191	0.0457	0.0661
+R-CE	<b>0.0516</b>	<b>0.0877</b>	0.0304	<b>0.0412</b>	0.0439	0.0723	0.0253	0.0337	0.0623	0.1208	0.0457	0.0668
+DeCA	0.0486	0.0832	0.0286	0.0390	0.0419	0.0688	0.0242	0.0321	0.0616	0.1202	0.0446	0.0659
+BOD	0.0481	0.0821	0.0278	0.038	0.0388	0.0639	0.0225	0.0299	0.0647	0.1212	0.0455	0.0664
+DDRM	<b>0.0516*</b>	<b>0.0870*</b>	<b>0.0305*</b>	<b>0.0412*</b>	<b>0.0468*</b>	<b>0.0742*</b>	<b>0.0273*</b>	<b>0.0355*</b>	<b>0.0667*</b>	<b>0.1221*</b>	<b>0.0508*</b>	<b>0.0710*</b>
SGL	0.0485	0.0835	0.0287	0.0393	0.0467	0.0758	0.0267	0.0353	0.0620	0.1164	0.0448	0.0648
+T-CE	0.0493	0.0840	0.0293	0.0398	0.0483	0.0765	0.0276	0.0361	0.0647	0.1184	0.0470	0.0667
+R-CE	0.0488	0.0831	0.0289	0.0393	0.0498	0.0772	0.0283	0.0367	0.0651	0.1165	0.0479	0.0670
+DeCA	0.0476	0.0801	0.0282	0.0380	0.0489	0.0764	0.0285	0.0368	0.0641	0.1183	0.0475	0.0673
+BOD	0.0505	0.0838	0.0303	0.0403	0.0517	0.0801	0.0300	0.0385	0.065	0.1234	0.0458	0.0668
+DDRM	<b>0.0517*</b>	<b>0.0860*</b>	<b>0.0312*</b>	<b>0.0415*</b>	<b>0.0535*</b>	<b>0.0813*</b>	<b>0.0313*</b>	<b>0.0396*</b>	<b>0.0698*</b>	<b>0.1261*</b>	<b>0.0530*</b>	<b>0.0739*</b>

submitted by users. For each dataset, the interactions with ratings  $< 4$  are regarded as false-positive interactions.

Following [39], we first arrange the user-item interactions chronologically based on the timestamps, and then split true-positive interactions (ratings  $\geq 4$ ) into training, validation and testing sets with a ratio of 7:1:2. To evaluate the effectiveness of denoising implicit feedback, we train and validate the framework on noisy interactions (both true-positive and false-positive interactions), and test the framework only on true-positive interactions. Specifically, we explore two types of noisy settings: natural noise and random noise. While keeping the testing set containing only true-positive interactions, 1) **Natural noise setting** introduces false-positive interactions (ratings  $< 4$ ) into the original training and validation sets; 2) **Random noise setting** randomly samples unobserved interactions into the original training and validation sets. Moreover, we ensure that the training and validation sets under the two noisy settings are at the same scale as the original dataset partition. The statistics of datasets are shown in Table 1.

**4.1.2 Baselines.** To demonstrate the efficacy of our proposed DDRM in denoising implicit feedback, we compare DDRM with the state-of-the-art model-agnostic denoising methods. In particular, 1) **R-CE**<sup>5</sup> [38] assumes large-loss interactions are more likely to be noisy and allocates lower weights to them. 2) **T-CE** [38], guided by the same assumption as R-CE, directly eliminates large-loss interactions using a dynamic threshold. 3) **DeCA**<sup>6</sup> [41] leverages predictions from different models as denoising signals, under the assumption that different models give more consistent predictions for clean data than for noisy data. 4) **BOD**<sup>7</sup> [43]

employs autoencoder-based generator to learn instance weight, thus denoising the data from data cleaning perspective.

Furthermore, we also compare DDRM with other competitive baselines including model perspective denoising methods and generative methods: 5) **AdaGCL**<sup>8</sup> [17] is a graph collaborative filtering-based denoising method. 6) **CDAE**<sup>9</sup> [48] introduces random noises to users' interactions during training and employs an auto-encoder for denoising. 7) **MultiVAE**<sup>10</sup> [23] utilizes variational auto-encoders with multinomial likelihood to model implicit feedback. 8) **DiffRec**<sup>11</sup> [39] is a diffusion-based generative recommender model that infers users' preferences by modeling the interaction probabilities in a denoising manner.

We implement DDRM and the aforementioned model-agnostic baselines to three representative backend models. 9) **MFBPR**<sup>12</sup> [30] is a collaborative filtering method based on matrix factorization with BPR ranking loss. 10) **LightGCN**<sup>13</sup> [13] leverages high-order neighbors information to enhance the user and item representations. 11) **SGL**<sup>14</sup> [45] is a self-supervised learning method, which conducts graph data augmentation for robust representation learning.

**Evaluation Metrics.** We adopt the full-ranking protocol to evaluate the top-K recommendation performance using two widely used metrics: Recall@K and NDCG@K with  $K = \{10, 20\}$ .

**4.1.3 Hyper-parameter Settings.** We fix the embedding size at 64 to maintain fairness when evaluating different methods. For model-agnostic methods, we initially determine the optimal hyper-parameters of the three backend models according to their default

<sup>5</sup><https://github.com/WenjieWWJ/DenoisingRec>

<sup>6</sup><https://github.com/wangyu-ustc/DeCA>

<sup>7</sup><https://github.com/CoderWZW/BOD>

<sup>8</sup><https://github.com/HKUDS/AdaGCL>

<sup>9</sup><https://github.com/henry0312/CDAE>

<sup>10</sup>[https://github.com/dawenl/vae\\_cf](https://github.com/dawenl/vae_cf)

<sup>11</sup><https://github.com/YiyanXu/DiffRec>

<sup>12</sup><https://github.com/guoyang9/BPR-pytorch>

<sup>13</sup><https://github.com/gusye1234/LightGCN-PyTorch>

<sup>14</sup><https://github.com/wujcan/SGL-Torch>

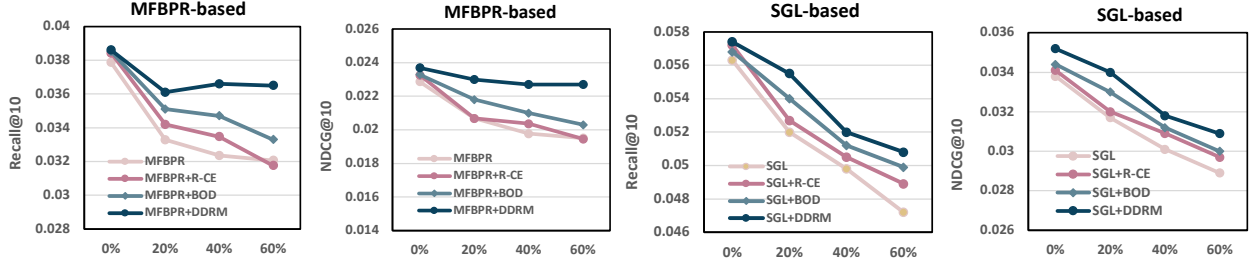


Figure 3: Performance comparison of noisy training with random noises in Yelp.

settings. Subsequently, we maintain the backend models’ hyper-parameters at their optimal and adjust only the specific denoising parameters, as per the original papers. For non-model-agnostic baselines, hyper-parameters are tuned within their default ranges.

Regarding our proposed DDRM, we have six hyper-parameters in total: the diffusion steps  $T$ , the noise lower bound  $\alpha_{\min}$ , the noise upper bound  $\alpha_{\max}$ , the noise scale  $s$  and loss balance factor  $\lambda$  and denoising weight factor  $\sigma$ . In detail,  $T$  is tuned within  $T = \{10, 20, \dots, 60\}$ . As for the noise-related parameters  $\alpha_{\min}$ ,  $\alpha_{\max}$  and  $s$ , we explore the combinations in  $\{1e-4, 1e-3\}$ ,  $\{1e-3, 1e-2\}$  and  $\{1e-4, 1e-3\}$ , respectively. For the loss-related parameters, loss balance factor  $\lambda$  and denoising weight factor  $\gamma$  are tuned within  $\{0.1, 0.2, \dots, 0.6\}$  and  $\{0, 0.05, 0.1, 0.2, \dots, 0.9\}$ , respectively.

## 4.2 Overall Performance (RQ1)

We conduct comprehensive experiments in natural noise setting to compare DDRM’s performance with other referenced baselines. The results, illustrated in Table 2, yield several key observations:

- DDRM mostly outperforms backend models and other model-agnostic denoising methods across all three datasets. This superior performance can be attributed to DDRM’s denoising diffusion process, which enhances robust representation learning through multi-step denoising.
- The performance of DeCA is not consistently better than the backend model. Two potential reasons emerge: 1) DeCA operates under the presumption that distinct models yield analogous predictions on clean data but deviate on noisy data. This assumption may not consistently hold true across our datasets. 2) DeCA’s training process involves four models optimized concurrently, which potentially induces instability.
- DiffRec consistently exhibits commendable performance across all three datasets, thereby highlighting the adeptness of diffusion models in denoising. DDRM, more flexible than DiffRec owing to its model-agnostic identity, can be deployed on any recommender model with user and item embeddings. What’s more, as for DiffRec, it requires to perform prediction tasks on all candidate items for a given user, resulting in high computational costs. DDRM only necessitates generating one single ideal item at the embedding level, and calculating the score between generated item embeddings and candidate item embeddings, which is more efficient. Furthermore, DDRM bolsters suboptimal models to perform comparably with, or even surpass DiffRec, thereby denoting its tangible enhancements upon the backend model.

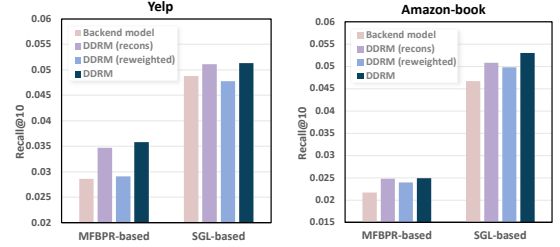


Figure 4: Contributions of reconstruction loss and reweighted loss to DDRM compared with backend models.

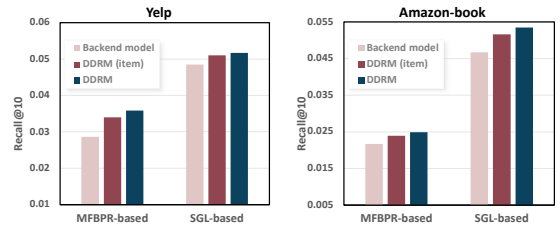


Figure 5: Contributions of user and item denoising modules to DDRM compared with backend models.

## 4.3 In-depth Analysis

**4.3.1 Random Noisy Training (RQ1).** We conduct random noisy training to evaluate the noise resistance capability of DDRM, comparing it with the two most competitive model-agnostic methods, R-CE and BOD, along with the backend model. The proportion of noise in our training settings spanned from 0% to 60%. We report the results in Figure 3. Similar results are seen with Amazon-book and ML-1M, but figures are omitted for brevity. The results show that: 1) As the noise ratio increases, there is an overall declining trend in the performance of the backend model, R-CE, BOD, and DDRM. This decline is attributed to the intensifying corruption of data due to the escalating noise level, making it challenging to discern genuine user preferences. 2) DDRM consistently outperforms both the backend model and R-CE in different noise ratio settings. This emphasizes DDRM’s commendable noise resistance, attributed to its robust representation learned through the diffusion process.

**4.3.2 Ablation Study (RQ2).** We execute ablation study to analyze DDRM from two distinct angles: loss perspective and module perspective.

• **Loss Perspective.** We assess the distinct contributions of the reconstruction loss from the reverse process and the reweighted loss from the extension, with outcomes depicted in Figure 4 for Yelp and Amazon-book (omitting ML-1M due to similar trends). We select

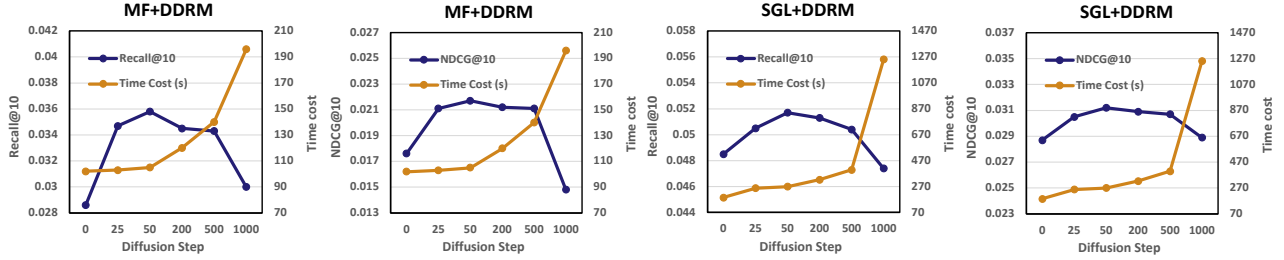


Figure 6: Performance and inference time cost comparison of using different diffusion steps in the DDRM in Yelp.

Table 3: Performance of different design variations in Yelp. Bold signifies the best performance among listed methods.

Model	R@10	R@20	N@10	N@20
SGL	0.0488	0.0841	0.0290	0.0397
DDRM	<b>0.0517</b>	<b>0.0860</b>	<b>0.0312</b>	<b>0.0415</b>
DDRM (Transformer)	0.0469	0.0786	0.0282	0.0378
DDRM (Noise Inference)	0.0213	0.0364	0.0133	0.0179
DDRM (Schedule: Linear)	0.0496	0.0834	0.0303	0.0404
DDRM (Schedule: Cosine)	0.0502	0.0834	0.0302	0.0406
DDRM (Schedule: Binomial)	0.0491	0.0828	0.0295	0.0397

the classic model MF and the graph model SGL which has uniformly commendable performance as our backend models. Based on these results, we can find that: 1) DDRM with reconstruction loss consistently outperforms the backend model, underscoring the efficacy of embedding denoising in DDRM. 2) DDRM with reweighted loss demonstrates overall effectiveness, yet its performance exhibits variability across different datasets and backend models. Notably, there exist instances where it falls short of the backend model’s performance. Furthermore, while improvements are observed in most cases, they are generally less substantial compared to those achieved by the DDRM with reconstruction loss. This suggests that the underlying assumptions of data cleaning methods (*cf.* Sec 1) may not always hold true. 3) Overall, while reconstruction and reweighted losses jointly contribute to DDRM’s effectiveness, the latter’s contributions are notably milder than those of the former.

• **Module Perspective.** To explore the impact of user denoising module and item denoising module, we conduct additional experiments on the Yelp and Amazon-book dataset only keeping the item denoising module of DDRM, with performance shown in Figure 5. It is important to note that experiments focusing solely on user denoising were not feasible in our work since we need to denoise item embeddings to generate ideal items for users in the inference phase. The results indicate a performance decrease compared to the full DDRM approach, suggesting that noise is present in user embeddings and necessitates user denoising. However, the performance with only item denoising still surpasses that of the backend models alone, which implies that the item denoising module contributes effectively to noise reduction. Therefore, it can be inferred that noise exists in both user and item representations, which prove the effectiveness and necessity of both user denoising module and item denoising module.

4.3.3 **Design Variations (RQ3).** We explore various design variations to validate the performance of our proposed DDRM. The performance of different designs is shown in Table 3, noting that the default backend model is SGL since it has uniformly competitive

performance across three datasets. The detailed analyses of each backbone design are provided below.

• **Architecture.** In this variation, we substitute the reconstruction MLP within the denoising module with the standard Transformer architecture. Our objective is to investigate whether the inclusion of a more intricate attention mechanism could bolster performance. Specifically, the collaborative information and step embedding are designated as the query and key in the Transformer, respectively, while the embedding requiring denoising is assigned as the value [34]. However, the results reveals the Transformer’s inferiority to the MLP in this context. A likely rationale for this outcome lies in the inductive bias introduced by the additional model structures in the Transformer, which is inappropriate in the denoising context, thus rendering suboptimal performance in reconstructing these embeddings. In contrast, the simplicity of the MLP proves to be highly efficient for the task at hand.

• **Noise Inference.** As inspired by [22], another approach we explored is denoising the embedding exclusively from pure noise instead of noisy average embeddings  $\bar{e}_T^i$  (*cf.* Section 3.3) during the inference phase. This method highly depends on diffusion’s impressive generation capabilities. Nonetheless, this leads to a substantial decline in performance, corroborating the assertions made in Section 1. Using pure noise as the starting point will increase the difficulty of the denoising process, primarily due to the absence of personalization which is crucial for regulating the generation of the ideal item in the recommendation scenario.

• **Noise Schedule.** We also examine the effect of using various noise schedules in the forward process of DDRM. Different types of noise schedules control how to add noise iteratively to the embeddings in the forward process, significantly influencing the denoising process. While the *linear variance* schedule is chosen for DDRM, we also experimented with three alternatives: the *linear*, *cosine*, and *binomial* schedules [14]. The findings in Table 3 revealed that the *linear variance* schedule surpasses the others. Its superiority lies in facilitating a smoother and more gradual noise injection in the embeddings, which improves the stability in the training phase compared with the *cosine* and *binomial* schedules. Furthermore, this schedule controls the maximum noise level injected into embeddings, which helps to preserve essential personalized information after the forward process compared with the *linear* schedule.

4.3.4 **Inference Step Analysis (RQ3).** We also delve into the impact of the number of diffusion steps on performance and time efficiency during inference, as depicted in Figure 6. The findings



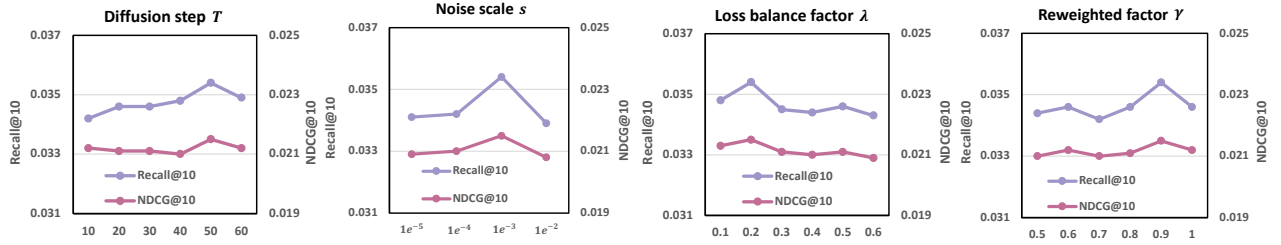


Figure 7: Hyper-parameter analysis in MFBPR-based DDRM for Yelp dataset.

indicate that optimal performance can be achieved with fewer than 100 diffusion steps. This contrasts with traditional diffusion models commonly employed in generative tasks, which often require up to 1000 steps. This is because these traditional models aim to reverse from pure noise during the inference, which need more steps to reverse in order to get high-quality image. DDRM, however, starts from embeddings that already contain personalized information which reflect users’ preference. Therefore, DDRM needs fewer steps to reverse compared with the traditional diffusion models, which saves the computation and time cost.

**4.3.5 Hyper-parameters Analysis.** For a more nuanced understanding, we select certain sensitive hyper-parameters, adjusting them within the ranges delineated in Section 4.1. The outcomes of these experiments are visually represented in Figure 7. From our observations: 1) With an increase in the diffusion step  $T$  and noise scale  $s$ , DDRM’s performance initially rises. This is attributed to an enhanced noise diversity in the data, enabling the model to foster more robust representations. Nonetheless, overly extending the diffusion steps and noise scale adversely affects performance, compromising the model’s personalization capabilities. Hence, it becomes imperative to judiciously determine the optimal diffusion step and noise scale to harness peak performance. 2) The loss balance factor  $\lambda$  influences DDRM performance by mediating the focus between recommendation and reconstruction tasks. While an increase in  $\lambda$  initially bolsters performance by prioritizing embedding denoising, too high a value risks neglecting the core recommendation task, undermining overall performance. 3) Appropriate selection of  $\gamma$  is crucial for effective denoising as higher values may filter out clean samples, while lower values may not comprehensively filter noisy samples.

## 5 RELATED WORK

• **Denosing Implicit Feedback.** Addressing the noise in implicit feedback, caused by false-positive interactions, has seen approaches primarily from data cleaning perspective [9, 38, 41, 43] and model perspective [17, 48]. Data cleaning methods often rely on specific assumptions to directly eliminate noisy interactions (e.g., WBPR [9]) or assign lower weights to such samples (e.g., T-CE [38], R-CE [38], DeCA [41]), resulting in limited adaptability across different backend models and datasets. Model perspective methods, on the other hand, focus on enhancing the noise resistance of recommender model. Specifically, some graph-based models (e.g., SGL [45], AdaGCL [17]) employ data augmentation to enhance data diversity and then regulate models to learn more robust representation. Additionally, some auto-encoder based models (e.g., CDAE [48])

intentionally corrupt users’ interactions by introducing different types of noise during training, and then attempt to reconstruct the original clean data using simple auto-encoders. However, these model perspective methods expect models to capture the complex noise distribution in the real world, which might pose a considerable challenge for the neural networks.

• **Generative Recommendation.** Generative models, notably Generative Adversarial Networks (GANs) [10, 19, 37] and Variational Autoencoders (VAEs) [23, 28, 55], are pivotal for personalized recommendations but face structural limitations [20, 32]. Emergent diffusion models, providing enhanced stability and representation compared with GANs and VAEs, have been recently explored in recommendation contexts [3, 18, 27, 46, 47]. While models like CODIGEM [36] and DiffRec [39] utilize diffusion models for inferring user preferences by modeling the distribution of users’ interaction probabilities, other research [7, 22, 25, 40] targets content generation at the embedding level, akin to our DDRM. For example, DiffuRec [22] and CDDRec [40] corrupt target item representations into pure noise in the forward process, subsequently reconstructing them condition on users’ historical interaction sequences. DiffuASR [25] uses diffusion models to generate new item sequences, mitigating data sparsity issues. However, these methods primarily concentrate on sequential recommendation and tend to overlook the presence of natural false-positive interactions in implicit feedback. In contrast, our proposed DDRM employs diffusion models to denoise implicit feedback, contributing to more robust representation learning.

## 6 CONCLUSION AND FUTURE WORK

In this work, we proposed the Denosing Diffusion Recommender Model (DDR), a plug-in model to bolster robust representation learning amidst noisy feedback for existing recommender models. Given user and item embeddings from any recommender models, DDR proactively injects Gaussian noises into the embeddings and then iteratively removes noise in the reverse process. To guide the reverse process, we designed a denoising module to encode collaborative information as guidance. During the inference phase, we utilized the average embeddings of users’ historically liked items as the starting point to generate an ideal item embedding, and then ground this embedding to existing item candidates as the recommendation for a user. Extensive experiment results demonstrate the superiority of DDR compared with other competitive baselines.

Future enhancements for DDR may involve: 1) enriching the denoising module with more well-designed neural networks, and 2) exploring adaptations of DDR for a broader range of recommendation tasks, such as sequential and bundle recommendations.

## REFERENCES

- [1] Huiyuan Chen, Yusan Lin, Menghai Pan, Lan Wang, Chin-Chia Michael Yeh, Xiaoting Li, Yan Zheng, Fei Wang, and Hao Yang. 2022. Denoising self-attentive sequential recommendation. In *RecSys*. ACM, 92–101.
- [2] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to debias for recommendation. In *SIGIR*. ACM, 21–30.
- [3] Lijian Chen, Wei Yuan, Tong Chen, Quoc Viet Hung Nguyen, Lizhen Cui, and Hongzhi Yin. 2023. Adversarial Item Promotion on Visually-Aware Recommender Systems by Guided Diffusion. *arXiv:2312.15826* (2023).
- [4] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. 2022. Diffusion models in vision: A survey. *arXiv:2209.04747* (2022).
- [5] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. 2018. An improved sampler for bayesian personalized ranking by leveraging view data. In *WWW*. ACM, 13–14.
- [6] Jingtao Ding, Guanghui Yu, Xiangnan He, Fuli Feng, Yong Li, and Depeng Jin. 2019. Sampler design for bayesian personalized ranking by leveraging view data. *TKDE* (2019), 667–681.
- [7] Hanwen Du, Huanhuan Yuan, Zhen Huang, Pengpeng Zhao, and Xiaofang Zhou. 2023. Sequential Recommendation with Diffusion Models. *arXiv:2304.04541*.
- [8] Ziwei Fan, Ke Xu, Zhang Dong, Hao Peng, Jiawei Zhang, and Philip S Yu. 2023. Graph Collaborative Signals Denoising and Augmentation for Recommendation. In *SIGIR*. ACM, 2037–2041.
- [9] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Personalized Ranking for Non-Uniformly Sampled Items. In *KDDCUP*. JMLR, 231–247.
- [10] Min Gao, Junwei Zhang, Junliang Yu, Jundong Li, Junhao Wen, and Qingyu Xiong. 2021. Recommender systems based on generative adversarial networks: A problem-driven perspective. *Inf. Sci.* (2021), 1166–1185.
- [11] 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 *SIGIR*. ACM, 1412–1422.
- [12] Shuyu Guo, Shuo Zhang, Weiwei Sun, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. 2023. Towards explainable conversational recommender systems. In *SIGIR*. ACM, 2786–2795.
- [13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.
- [14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *NeurIPS*. Curran Associates, Inc., 6840–6851.
- [15] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. Curran Associates, Inc., 12454–12465.
- [16] Lei Huang, Hengtong Zhang, Tingyang Xu, and Ka-Chun Wong. 2023. Mdm: Molecular diffusion model for 3d molecule generation. In *AAAI*. AAAI press, 5105–5112.
- [17] Yangqin Jiang, Chao Huang, and Lianghao Huang. 2023. Adaptive Graph Contrastive Learning for Recommendation. In *KDD*. ACM, 4252–4261.
- [18] Yangqin Jiang, Yuhao Yang, Lianghao Xia, and Chao Huang. 2023. DiffKG: Knowledge Graph Diffusion Model for Recommendation. *arXiv:2312.16890* (2023).
- [19] Binbin Jin, Defu Lian, Zheng Liu, Qi Liu, Jianhui Ma, Xing Xie, and Enhong Chen. 2020. Sampling-decomposable generative adversarial recommender. In *NeurIPS*. Curran Associates, Inc., 22629–22639.
- [20] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *NeurIPS*. Curran Associates, Inc., 4743–4751.
- [21] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tat-Sunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *arXiv:2205.14217*.
- [22] Zihao Li, Aixin Sun, and Chenliang Li. 2023. DiffuRec: A Diffusion Model for Sequential Recommendation. *arXiv:2304.00686*.
- [23] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW*. ACM, 689–698.
- [24] Weilin Lin, Xiangyu Zhao, Yejing Wang, Yuanshao Zhu, and Wanyu Wang. 2023. AutoDenoise: Automatic Data Instance Denoising for Recommendations. In *WWW*. ACM, 1003–1011.
- [25] Qidong Liu, Fan Yan, Xiangyu Zhao, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Feng Tian. 2023. Diffusion Augmentation for Sequential Recommendation. *arXiv:2309.12858*.
- [26] Calvin Luo. 2022. Understanding diffusion models: A unified perspective. *arXiv:2208.11970*.
- [27] Haokai Ma, Ruobing Xie, Lei Meng, Xin Chen, Xu Zhang, Leyu Lin, and Zhanhui Kang. 2024. Plug-in Diffusion Model for Sequential Recommendation. *arXiv:2401.02913* (2024).
- [28] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning Disentangled Representations for Recommendation. In *NeurIPS*. Curran Associates, Inc., 5712–5723.
- [29] Yuhuan Quan, Jingtao Ding, Chen Gao, Lingling Yi, Depeng Jin, and Yong Li. 2023. Robust Preference-Guided Denoising for Graph based Social Recommendation. In *WWW*. ACM, 1097–1108.
- [30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. AUAI Press, 452–461.
- [31] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*. IEEE, 22500–22510.
- [32] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*. PMLR, 2256–2265.
- [33] Changxin Tian, Yuexiang Xie, Yaliang Li, Nan Yang, and Wayne Xin Zhao. 2022. Learning to denoise unreliable interactions for graph collaborative filtering. In *SIGIR*. ACM, 122–132.
- [34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [35] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. 2022. Digress: Discrete denoising diffusion for graph generation. In *ICLR*.
- [36] Joojo Walker, Ting Zhong, Fengli Zhang, Qiang Gao, and Fan Zhou. 2022. Recommendation via Collaborative Diffusion Generative Model. In *KSEM*. Springer, 593–605.
- [37] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minmax game for unifying generative and discriminative information retrieval models. In *SIGIR*. ACM, 515–524.
- [38] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. In *WSDM*. ACM, 373–381.
- [39] Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. 2023. Diffusion Recommender Model. In *SIGIR*. ACM, 832–841.
- [40] Yu Wang, Zhiwei Liu, Liangwei Yang, and Philip S. Yu. 2023. Conditional Denoising Diffusion for Sequential Recommendation. *arXiv:2304.11433*.
- [41] Yu Wang, Xin Xin, Zaiqiao Meng, Joemon M Jose, Fuli Feng, and Xiangnan He. 2022. Learning Robust Recommenders through Cross-Model Agreement. In *WWW*. ACM, 2015–2025.
- [42] Zhenlei Wang and Xu Chen. 2023. Robust Recommendation with Adversarial Gaussian Data Augmentation. In *WWW*. ACM, 897–905.
- [43] Zongwei Wang, Min Gao, Wentao Li, Junliang Yu, Linxin Guo, and Hongzhi Yin. 2023. Efficient Bi-Level Optimization for Recommendation Denoising. In *SIGKDD*. ACM, 2502–2511.
- [44] 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 *MM*. ACM, 3070–3078.
- [45] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-Supervised Graph Learning for Recommendation. In *SIGIR*. ACM, 726–735.
- [46] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2022. DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation. *TKDE* 34, 10 (2022), 4753–4766.
- [47] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *SIGIR*. 235–244.
- [48] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. ACM, 153–162.
- [49] Xin Xin, Xiangyuan Liu, Hanbing Wang, Pengjie Ren, Zhumin Chen, Jiahuan Lei, Xinlei Shi, Hengliang Luo, Joemon M Jose, Maarten de Rijke, et al. 2023. Improving Implicit Feedback-Based Recommendation through Multi-Behavior Alignment. In *SIGIR*. ACM, 932–941.
- [50] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2022. Diffusion models: A comprehensive survey of methods and applications. *Comput. Surveys* (2022).
- [51] Wenhui Yu and Zheng Qin. 2020. Sampler design for implicit feedback data by noisy-label robust learning. In *SIGIR*. ACM, 861–870.
- [52] Wei Yuan, Chaoqun Yang, Quoc Viet Hung Nguyen, Lizhen Cui, Tieke He, and Hongzhi Yin. 2023. Interaction-level membership inference attack against federated recommender systems. In *WWW*. ACM, 1053–1062.
- [53] Chi Zhang, Rui Chen, Xiangyu Zhao, Qilong Han, and Li Li. 2023. Denoising and Prompt-Tuning for Multi-Behavior Recommendation. In *WWW*. ACM, 1355–1363.
- [54] 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. *TOIS* (2023).
- [55] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. Autosvd++ an efficient hybrid collaborative filtering model via contractive auto-encoders. In *SIGIR*. ACM, 957–960.
- [56] Jujia Zhao, Wenjie Wang, Xinyu Lin, Leigang Qu, Jizhi Zhang, and Tat-Seng Chua. 2023. Popularity-aware Distributionally Robust Optimization for Recommendation System. In *CIKM*. ACM, 4967–4973.