Diffusion Recommender Model

Wenjie Wang
wenjiewang96@gmail.com
National University of Singapore

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

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

Xinyu Lin
xylin1028@gmail.com
National University of Singapore

Xiangnan He
xiangnanhe@gmail.com
University of Science and Technology of China

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

ABSTRACT

Generative models such as Generative Adversarial Networks (GANs) and Variational Auto-Encoders (VAEs) are widely utilized to model the generative process of user interactions. However, they suffer from intrinsic limitations such as the instability of GANs and the restricted representation ability of VAEs. Such limitations hinder the accurate modeling of the complex user interaction generation procedure, such as noisy interactions caused by various interference factors. In light of the impressive advantages of *Diffusion Models* (DMs) over traditional generative models in image synthesis, we propose a novel ***Diff**usion **Rec**ommender Model* (named DiffRec) to learn the generative process in a denoising manner. To retain personalized information in user interactions, DiffRec reduces the added noises and avoids corrupting users' interactions into pure noises like in image synthesis. In addition, we extend traditional DMs to tackle the unique challenges in recommendation: high resource costs for large-scale item prediction and temporal shifts of user preference. To this end, we propose two extensions of DiffRec: L-DiffRec clusters items for dimension compression and conducts the diffusion processes in the latent space; and T-DiffRec reweights user interactions based on the interaction timestamps to encode temporal information. We conduct extensive experiments on three datasets under multiple settings (*e.g.,* clean training, noisy training, and temporal training). The empirical results validate the superiority of DiffRec with two extensions over competitive baselines.

CCS CONCEPTS

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

KEYWORDS

Generative Recommender Model, Diffusion Model, Latent and Temporal Diffusion Recommender Models

*Corresponding author: Fuli Feng. This research is supported by the National Key Research and Development Program of China (2020YFB1406703), the National Natural Science Foundation of China (62272437), and Huawei International Pte Ltd.

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591663

ACM Reference Format:
Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. 2023. Diffusion Recommender Model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3539618.3591663

1 INTRODUCTION

Generative models such as Generative Adversarial Networks (GANs) and Variational Auto-Encoders (VAEs) have been broadly utilized for personalized recommendation [19, 37, 49]. Generally speaking, generative recommender models learn the generative process to infer the user interaction probabilities over all non-interacted items. Such generative process typically assumes that users' interaction behaviors with items (*e.g.,* clicks) are determined by some latent factors (*e.g.,* user preference). Due to aligning with the real-world interaction generation procedure, generative recommender models have achieved significant success [19, 37].

Generative recommender models mainly fall into two groups:

- GAN-based models utilize a generator to estimate users' interaction probabilities and leverage adversarial training to optimize the parameters [13, 37]. However, adversarial training is typically unstable, leading to unsatisfactory performance.
- VAEs-based models use an encoder to approximate the posterior distribution over latent factors and maximize the likelihood of observed interactions (Figure 1(a)) [19, 24]. While VAEs typically outperform GANs in recommendation, VAEs suffer from the trade-off between tractability and representation ability [14, 34]. Tractable and simple encoders might not well capture heterogeneous user preference while the posterior distribution of complex models is likely to be intractable [34].

Diffusion Models (DMs) [10, 34] have achieved state-of-the-art results in image synthesis tasks [31], which alleviate the trade-off by gradually corrupting the images in a tractable forward process and learning the reverse reconstruction iteratively. As shown in Figure 1(b), DMs forwardly corrupt $\boldsymbol{x}_0$ with random noises step by step, and recover $\boldsymbol{x}_0$ from corrupted $\boldsymbol{x}_T$ iteratively. This forward process leads to a tractable posterior [34], and also opens the door to iteratively modeling complex distributions by flexible neural networks in the reverse generation. The objectives of recommender models align well with DMs since recommender models essentially infer the future interaction probabilities based on corrupted historical interactions (Figure 1(c)), where corruption implies that the interactions are noisy due to false-positive and false-negative items [32, 38]. As such, exploring DMs for recommendation has

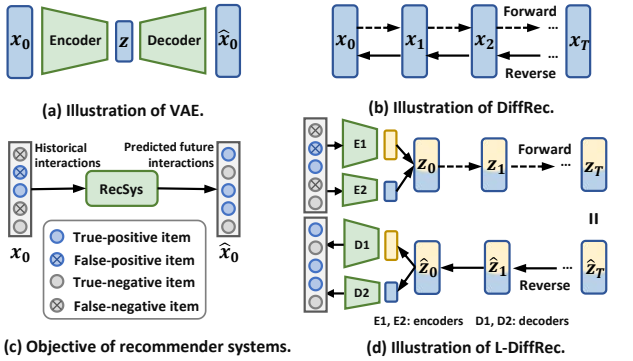

Figure 1: Illustration of VAE, DiffRec, the objective of recommender systems, and L-DiffRec.

great potential to model the complex interaction generation more accurately with strong representation ability.

We propose a ***Diff**usion **Rec**ommender Model* named DiffRec, which infers users' interaction probabilities in a denoising manner. Technically, DiffRec gradually corrupts users' interaction histories by injecting scheduled Gaussian noises in the forward process, and then recovers original interactions from the corrupted interactions iteratively via a parameterized neural network. Nevertheless, we cannot directly graft the forward process in the image domain due to the necessity of generating personalized recommendations. To retain personalized information in users' corrupted interactions, we should avoid corrupting users' interaction histories into pure noises like in image synthesis. We thus significantly decrease the added noise scales in the forward process (see Section 3.4).

Taking one step further, we handle two essential challenges in building generative models for recommendation: large-scale item prediction and temporal modeling. In detail, 1) generative models require extensive resource costs as predicting the interaction probabilities of all items simultaneously [19], limiting their application to large-scale item recommendation; and 2) generative models have to capture the temporal information in the interaction sequence, which is crucial for handling user preference shifts [47]. To this end, we further extend DiffRec to *Latent DiffRec* (named L-DiffRec) and *Temporal DiffRec* (named T-DiffRec).

- L-DiffRec clusters items into groups, compresses the interaction vector over each group into a low-dimensional latent vector via a group-specific VAE, and conducts the forward and reverse diffusion processes in the latent space (Figure 1(d)). Owing to the clustering and latent diffusion, L-DiffRec significantly reduces the model parameters and memory costs, enhancing the ability of large-scale item prediction (see Section 3.5 and 4.3).
- T-DiffRec models the interaction sequence via a simple yet effective time-aware reweighting strategy. Intuitively, users' later interactions are assigned with larger weights, and then fed into DiffRec for training and inference (see Section 3.6 and 4.4).

We conduct extensive experiments on three representative datasets and compare DiffRec with various baselines under multiple settings (*e.g.,* clean training, noisy training with natural or random noises, and temporal training), validating the superiority of our proposed DiffRec and two extensions. We release our code and data at https://github.com/YiyanXu/DiffRec.

To sum up, the contributions of this work are as follows.

- We propose a novel Diffusion Recommender Model, a totally new recommender paradigm that points out a promising future direction for generative recommender models.
- We extend conventional Diffusion Models to reduce the resource costs for high-dimensional categorical predictions and enable the time-sensitive modeling of interaction sequences.
- We conduct substantial experiments on three datasets under various settings, demonstrating remarkable improvements of DiffRec with two extensions over the baselines.

2 PRELIMINARY

DMs have achieved impressive success in various fields, mainly consisting of forward and reverse processes [10, 34].

• **Forward process.** Given an input data sample $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, the forward process constructs the latent variables $\boldsymbol{x}_{1:T}$ in a Markov chain by gradually adding Gaussian noises in T steps. Specifically, DMs define the forward transition $\boldsymbol{x}_{t-1} \rightarrow \boldsymbol{x}_t$ as $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I})$, where $t \in \{1, \dots, T\}$ refers to the diffusion step, $\mathcal{N}$ denotes the Gaussian distribution, and $\beta_t \in (0, 1)$ controls the noise scales added at the step t. If $T \rightarrow \infty$, $\boldsymbol{x}_T$ approaches a standard Gaussian distribution [10].

• **Reverse process.** DMs learn to remove the added noises from $\boldsymbol{x}_t$ to recover $\boldsymbol{x}_{t-1}$ in the reverse step, aiming to capture minor changes in the complex generation process. Formally, taking $\boldsymbol{x}_T$ as the initial state, DMs learn the denoising process $\boldsymbol{x}_t \rightarrow \boldsymbol{x}_{t-1}$ iteratively by $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \Sigma_\theta(\boldsymbol{x}_t, t))$, where $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ and $\Sigma_\theta(\boldsymbol{x}_t, t)$ are the mean and covariance of the Gaussian distribution predicted by a neural network with parameters θ.

• **Optimization.** DMs are optimized by maximizing the Evidence Lower Bound (ELBO) of the likelihood of observed input data $\boldsymbol{x}_0$:

$$
\begin{aligned}
\log p(\boldsymbol{x}_0) &= \log \int p(\boldsymbol{x}_{0:T}) \mathrm{d}\boldsymbol{x}_{1:T} \\
&= \log \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[\frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \right] \\
&\geq \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)} \left[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{(reconstruction term)}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{(prior matching term)}} \\
&\quad - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{(denoising matching term)}},
\end{aligned} \tag{1}
$$

where 1) the reconstruction term denotes the negative reconstruction error over $\boldsymbol{x}_0$; 2) the prior matching term is a constant without trainable parameters and thus ignorable in the optimization; and 3) the denoising matching terms regulate $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to align with the tractable ground-truth transition step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ [23]. In this way, θ is optimized to iteratively recover $\boldsymbol{x}_{t-1}$ from $\boldsymbol{x}_t$. According to [10], the denoising matching terms can be simplified as $\sum_{t=2}^{T} \mathbb{E}_{t,\boldsymbol{\epsilon}} \left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2 \right]$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$; and $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is parameterized by a neural network (*e.g.,* U-Net [10]) to predict the noises $\boldsymbol{\epsilon}$ that determine $\boldsymbol{x}_t$ from $\boldsymbol{x}_0$ in the forward process [23].

• **Inference.** After training θ, DMs can draw $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and leverage $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to iteratively repeat the generation process $\boldsymbol{x}_T \rightarrow \boldsymbol{x}_{T-1} \rightarrow \cdots \rightarrow \boldsymbol{x}_0$. Besides, prior studies consider adding some conditions to realize the controllable generation [18, 31].

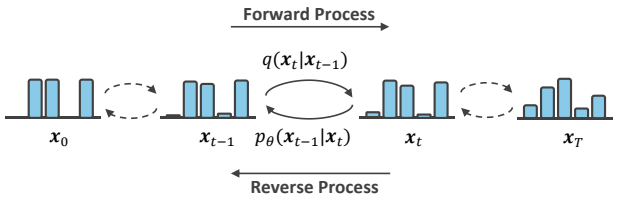

Figure 2: An overview of DiffRec, where the histogram denotes the corrupted interactions of a user over all items. The forward process gradually corrupts the user's interaction history by the transition step $q(x_t|x_{t-1})$, and then the model learns to recover x_0 using $p_\theta(x_{t-1}|x_t)$ step by step.

3 DIFFUSION RECOMMENDER MODEL

To take advantage of the strong generation ability of DMs, we propose a novel DiffRec to predict users' future interaction probabilities from corrupted interactions. Given users' historical interactions, DiffRec gradually corrupts them by adding noises in a forward process, and then learns to recover original interactions iteratively. By such iterative denoising training, DiffRec can model complex interaction generation procedures and mitigate the effects of noisy interactions. Eventually, the recovered interaction probabilities are used to rank and recommend non-interacted items. In addition, we present two extensions of DiffRec for large-scale item prediction and temporal modeling to facilitate the use of DiffRec in practical recommender systems.

3.1 Forward and Reverse Processes

As shown in Figure 2, DiffRec has two critical processes: 1) a forward process corrupts users' interaction histories by adding Gaussian noises step by step, and 2) a reverse process gradually learns to denoise and output the interaction probabilities.

• **Forward process.** Given a user u with the interaction history over an item set $\mathcal{I}$, *i.e.,* $\boldsymbol{x}_u = [x_u^1, x_u^2, \ldots, x_u^{|\mathcal{I}|}]$ where $x_u^i = 1$ or 0 implies whether user u has interacted with item i or not, we can set $\boldsymbol{x}_0 = \boldsymbol{x}_u$ as the initial state[1] and parameterize the transition by

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}), \tag{2}$$

where $\beta_t \in (0,1)$ controls the Gaussian noise scales added at each step t. Thanks to *the reparameterization trick* [10] and the additivity of two independent Gaussian noises [10, 23], we can directly obtain $\boldsymbol{x}_t$ from $\boldsymbol{x}_0$. Formally,

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I}), \tag{3}$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{t'=1}^{t} \alpha_{t'}$, and then we can reparameterize $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. To regulate the added noises in $\boldsymbol{x}_{1:T}$, we design a linear noise schedule for $1-\bar{\alpha}_t$, *i.e.,*

$$1 - \bar{\alpha}_t = s \cdot \left[\alpha_{\min} + \frac{t-1}{T-1}(\alpha_{\max} - \alpha_{\min})\right], \quad t \in \{1, \ldots, T\}, \tag{4}$$

where a hyper-parameter $s \in [0, 1]$ controls the noise scales, and two hyper-parameters $\alpha_{\min} < \alpha_{\max} \in (0, 1)$ indicating the upper and lower bounds of the added noises.

[1]For notation brevity, we omit the subscript u in $\boldsymbol{x}_0$ for user u.

• **Reverse process.** Starting from $\boldsymbol{x}_T$, the reverse process gradually recovers users' interactions by the denoising transition step:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \Sigma_\theta(\boldsymbol{x}_t, t)), \tag{5}$$

where $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ and $\Sigma_\theta(\boldsymbol{x}_t, t)$ are the Gaussian parameters outputted by any neural networks with learnable parameters θ.

3.2 DiffRec Training

To learn θ, DiffRec aims to maximize the ELBO of observed user interactions $\boldsymbol{x}_0$:

$$\begin{aligned}\log p(\boldsymbol{x}_0) \geq & \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{(reconstruction term)}} \\ & - \underbrace{\textstyle\sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{(denoising matching term)}}.\end{aligned} \tag{6}$$

Note that the prior matching term in Eq. (1) is omitted as it is a constant. Besides, the reconstruction term measures the recovery probability of $\boldsymbol{x}_0$ while denoising matching terms regulate the recovery of $\boldsymbol{x}_{t-1}$ with t varying from 2 to T in the reverse process. So far, the optimization lies in maximizing the reconstruction term and denoising matching terms.

• **Estimation of denoising matching terms.** The denoising matching term forces $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to approximate the tractable distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ via KL divergence. Through Bayes rules, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ can be rewritten as the following closed form [23]:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \propto \mathcal{N}(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0, t), \sigma^2(t)\boldsymbol{I}), \text{ where} \tag{7}$$

$$\begin{cases} \tilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0, t) = \dfrac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t + \dfrac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\boldsymbol{x}_0, \\ \sigma^2(t) = \dfrac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}. \end{cases} \tag{8}$$

$\tilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0, t)$ and $\sigma^2(t)\boldsymbol{I}$ are the mean and covariance of $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ derived from Eq. (2) and Eq. (3) [10]. Besides, to keep training stability and simplify the calculation, we ignore the learning of $\Sigma_\theta(\boldsymbol{x}_t, t)$ in $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ and directly set $\Sigma_\theta(\boldsymbol{x}_t, t) = \sigma^2(t)\boldsymbol{I}$ by following [10]. Thereafter, the denoising matching term $\mathcal{L}_t$ at step t can be calculated by

$$\begin{aligned}\mathcal{L}_t &\triangleq \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right] \\ &= \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\frac{1}{2\sigma^2(t)}\left[\parallel \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) - \tilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0, t) \parallel_2^2\right]\right],\end{aligned} \tag{9}$$

which pushes $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ to be close to $\tilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0, t)$. Following Eq. (8), we can similarly factorize $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ via

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_t, t), \tag{10}$$

where $\hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_t, t)$ is the predicted $\boldsymbol{x}_0$ based on $\boldsymbol{x}_t$ and t. Furthermore, by substituting Eq. (10) and Eq. (8) into Eq. (9), we have

$$\mathcal{L}_t = \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[\frac{1}{2}\left(\frac{\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}} - \frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}\right) \parallel \hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{x}_0 \parallel_2^2\right], \tag{11}$$

which regulates $\hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_t, t)$ to predict $\boldsymbol{x}_0$ accurately.

To summarize, for estimating denoising matching terms, we need to implement $\hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_t, t)$ by neural networks and calculate Eq. (11). Following MultiVAE [19], we also instantiate $\hat{\boldsymbol{x}}_\theta(\cdot)$ via a Multi-Layer Perceptron (MLP) that takes $\boldsymbol{x}_t$ and the step embedding of t as inputs to predict $\boldsymbol{x}_0$.

Algorithm 1 DiffRec Training

Input: all users' interactions $\bar{\boldsymbol{X}}$ and randomly initialized θ.

repeat
  Sample a batch of users' interactions $\boldsymbol{X} \subset \bar{\boldsymbol{X}}$.
  for all $\boldsymbol{x}_0 \in \boldsymbol{X}$ **do**
    Sample $t \sim \mathcal{U}(1, T)$ or $t \sim p_t$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$;
    Compute $\boldsymbol{x}_t$ given $\boldsymbol{x}_0$, t, and $\boldsymbol{\epsilon}$ via $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ in Eq. (3);
    Compute $\mathcal{L}_t$ by Eq. (11) if $t > 1$, otherwise by Eq. (12);
    Take gradient descent step on $\nabla_\theta \mathcal{L}_t$ to optimize θ;
until converged

Output: optimized θ.

• **Estimation of the reconstruction term.** We define $\mathcal{L}_1$ as the negative of the reconstruction term in Eq. (6), and calculate $\mathcal{L}_1$ by

$$\begin{aligned}\mathcal{L}_1 &\triangleq -\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)\right] \\ &= \mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\| \hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_1, 1) - \boldsymbol{x}_0 \|_2^2\right],\end{aligned} \tag{12}$$

where we estimate the Gaussian log-likelihood $\log p(\boldsymbol{x}_0|\boldsymbol{x}_1)$ by unweighted $-||\hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_1, 1) - \boldsymbol{x}_0||_2^2$ as discussed in [19].

• **Optimization.** According to Eq. (11) and Eq. (12), ELBO in Eq. (6) can be formulated as $-\mathcal{L}_1 - \sum_{t=2}^{T} \mathcal{L}_t$. Therefore, to maximize the ELBO, we can optimize θ in $\hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_t, t)$ by minimizing $\sum_{t=1}^{T} \mathcal{L}_t$. In the practical implementation, we uniformly sample step t to optimize an expectation $\mathcal{L}(\boldsymbol{x}_0, \theta)$ over $t \sim \mathcal{U}(1, T)$. Formally,

$$\mathcal{L}(\boldsymbol{x}_0, \theta) = \mathbb{E}_{t\sim\mathcal{U}(1,T)}\, \mathcal{L}_t. \tag{13}$$

The training procedure of DiffRec is presented in Algorithm 1.

• **Importance sampling.** Since the optimization difficulty might vary across different steps. we consider using *importance sampling* [25] to emphasize the learning over the steps with large loss values of $\mathcal{L}_t$. Formally, we use a new sampling strategy for t:

$$\mathcal{L}^{\triangle}(\boldsymbol{x}_0, \theta) = \mathbb{E}_{t\sim p_t}\left[\frac{\mathcal{L}_t}{p_t}\right], \tag{14}$$

where $p_t \propto \sqrt{\mathbb{E}\left[\mathcal{L}_t^2\right]} / \sqrt{\sum_{t'=1}^{T} \mathbb{E}\left[\mathcal{L}_{t'}^2\right]}$ denotes the sampling probability and $\sum_{t=1}^{T} p_t = 1$. We here calculate $\mathbb{E}\left[\mathcal{L}_t^2\right]$ by collecting ten $\mathcal{L}_t$ values during training and taking the average. Before acquiring enough $\mathcal{L}_t$, we still adopt the uniform sampling. Intuitively, the steps with large $\mathcal{L}_t$ values will be more easily sampled.

3.3 DiffRec Inference

In image synthesis tasks, DMs draw random Gaussian noises for reverse generation, possibly guided by the gradients from a pre-trained classifier or other signals such as textual queries. However, corrupting interactions into pure noises will hurt personalized user preference in recommendation (see empirical evidence in Section 4.2.3). It is also non-trivial to design additional classifiers or guidance signals. As such, we propose a simple inference strategy to align with DiffRec training for interaction prediction.

Specifically, DiffRec firstly corrupts $\boldsymbol{x}_0$ by $\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \cdots \to \boldsymbol{x}_{T'}$ for T' steps in the forward process, and then sets $\hat{\boldsymbol{x}}_T = \boldsymbol{x}_{T'}$ to execute reverse denoising $\hat{\boldsymbol{x}}_T \to \hat{\boldsymbol{x}}_{T-1} \to \cdots \to \hat{\boldsymbol{x}}_0$ for T steps. The reverse denoising ignores the variance (like in MultiVAE [19]) and utilize $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_\theta(\hat{\boldsymbol{x}}_t, t)$ via Eq. (10) for deterministic inference. In particular, in considering 1) the collected user interactions are naturally noisy due to false-positive and false-negative interactions [38, 39] and 2) retaining personalized information, we reduce the added noises in the forward process by setting $T' < T$. Finally, we use $\hat{\boldsymbol{x}}_0$ for item ranking and recommend top-ranked items. The inference procedure is summarized in Algorithm 2.

Algorithm 2 DiffRec Inference

Input: θ and the interaction history $\boldsymbol{x}_0$ of user u.

Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
Compute $\boldsymbol{x}_{T'}$ given $\boldsymbol{x}_0$, T', and $\boldsymbol{\epsilon}$ via Eq. (3), and set $\hat{\boldsymbol{x}}_T = \boldsymbol{x}_{T'}$.
for $t = T, \ldots, 1$ **do**
  $\hat{\boldsymbol{x}}_{t-1} = \boldsymbol{\mu}_\theta(\hat{\boldsymbol{x}}_t, t)$ calculated from $\hat{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{x}}_\theta(\cdot)$ via Eq. (10);

Output: the interaction probabilities $\hat{\boldsymbol{x}}_0$ for user u.

3.4 Discussion

Unlike image synthesis, we highlight two special points of DiffRec.

- **Personalized recommendation.** 1) During training, we do not corrupt users' interactions into pure noises for retaining some personalized information; that is, the latent variable $\boldsymbol{x}_T$ does not approach the standard Gaussian noises that lose extensive personalized characteristics. It is similar to the selection of β in MultiVAE to control the strength of the prior constraint, *i.e.,* the KL divergence (see Section 2.2.2 in [19]). In practice, We reduce s and $\alpha_{\max}$ in the noise schedule of Eq. (4) to lessen the noises. And 2) we also decrease the added noises for inference by controlling $T' < T$ by considering the natural noises in user interactions.
- **$\boldsymbol{x}_0$-ELBO.** DiffRec is optimized by predicting $\boldsymbol{x}_0$ instead of $\boldsymbol{\epsilon}$ like in Section 2 because: 1) the key objective of recommendation is to predict $\hat{\boldsymbol{x}}_0$ for item ranking, and thus $\boldsymbol{x}_0$-ELBO is intuitively more appropriate for our task; and 2) randomly sampled $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is unsteady and forcing an MLP to estimate such a $\boldsymbol{\epsilon}$ is more challenging (see empirical analysis in Section 4.2.3).

3.5 Latent Diffusion

Generative models, such as MultiVAE and DiffRec, predict the interaction probabilities $\hat{\boldsymbol{x}}_0$ over all items simultaneously, requiring extensive resources and limiting large-scale item prediction in industry. To reduce the costs, we offer L-DiffRec, which clusters items for dimension compression via multiple VAEs and conducts diffusion processes in the latent space as shown in Figure 3.

• **Encoding for compression.** Given an item set $\mathcal{I}$, L-DiffRec first adopts *k-means* to cluster items into C categories $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_C\}$ based on item representations (*e.g.,* trained item embeddings from LightGCN). L-DiffRec then divides user interaction vector $\boldsymbol{x}_0$ into C parts according to the clusters, *i.e.,* $\boldsymbol{x}_0 \to \{\boldsymbol{x}_0^c\}_{c=1}^{C}$, where $\boldsymbol{x}_0^c$ represents the interactions of user u over $\mathcal{I}_c$. Afterwards, we use a variational encoder parameterized by ϕ_c to compress each $\boldsymbol{x}_0^c$ to a low-dimensional vector $\boldsymbol{z}_0^c$, where the encoder predicts $\boldsymbol{\mu}_{\phi_c}$ and $\sigma_{\phi_c}^2 \boldsymbol{I}$ as the mean and covariance of the variational distribution $q_{\phi_c}(\boldsymbol{z}_0^c|\boldsymbol{x}_0^c) = \mathcal{N}(\boldsymbol{z}_0^c; \boldsymbol{\mu}_{\phi_c}(\boldsymbol{x}_0^c), \sigma_{\phi_c}^2(\boldsymbol{x}_0^c)\boldsymbol{I})$. The clustering can lessen resource costs since it can 1) achieve parallel calculation of different categories and 2) break the full connections among the multiple encoders to save parameters compared to vanilla VAE [19].

• **Latent diffusion.** By concatenating $\{\boldsymbol{z}_0^c\}_{c=1}^{C}$, we can obtain the compressed $\boldsymbol{z}_0$ for diffusion. Like DiffRec training, we replace $\boldsymbol{x}_0$

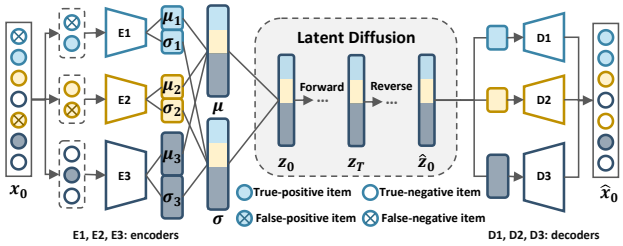

Figure 3: Illustration of L-DiffRec. $z_0 = \mu + \sigma \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. L-DiffRec clusters items for compression via multiple VAEs and conducts latent diffusion.

with z_0 to do the forward and reverse processes in the latent space. Similar to Eq. (13), we have the optimization loss as $\mathcal{L}(z_0, \theta) = \mathbb{E}_{t\sim\mathcal{U}(1,T)}\mathcal{L}_t$, where θ marks the parameters of the denoising MLP.

• **Decoding.** As shown in Figure 3, we split the reconstructed $\hat{z}_0$ from the reverse process into $\{\hat{z}_0^c\}_{c=1}^C$ according to item clusters. Each $\hat{z}_0^c$ is then fed into a separate decoder parameterized by ψ_c to predict $\hat{x}_0$ via $p_{\psi_c}(\hat{x}_0^c|\hat{z}_0^c)$, which is similar to MultiVAE [19].

• **Training.** Intuitively, the encoder q_{ϕ_c} and decoder p_{ψ_c} jointly constitute a VAE that bridges the interaction space and the latent space. Following MultiVAE [19], the set of VAEs with $\phi = \{\phi_c\}_{c=1}^C$ and $\psi = \{\psi_c\}_{c=1}^C$ could optimized by:

$$\mathcal{L}_v(\boldsymbol{x}_0, \phi, \psi) = \sum_{c=1}^{C} [\mathbb{E}_{q_{\phi_c}(\boldsymbol{z}_0^c|\boldsymbol{x}_0^c)} \left[\log p_{\psi_c}(\boldsymbol{x}_0^c|\boldsymbol{z}_0^c)\right] - \gamma \cdot D_{\mathrm{KL}}(q_{\phi_c}(\boldsymbol{z}_0^c|\boldsymbol{x}_0^c)||p(\boldsymbol{z}_0^c))], \tag{15}$$

where γ is to control the strength of KL regularization. Subsequently, combining the loss of diffusion and VAEs, we have $\mathcal{L}_v(\boldsymbol{x}_0, \phi, \psi) + \lambda \cdot \mathcal{L}(z_0, \theta)$ for L-DiffRec optimization, where the hyper-parameter λ ensures the two terms in the same magnitude.

• **Inference.** For inference, L-DiffRec first splits $\boldsymbol{x}_0$ into $\{\boldsymbol{x}_0^c\}_{c=1}^C$, and then compresses each $\boldsymbol{x}_0^c$ into a deterministic variable $\boldsymbol{z}_0^c = \boldsymbol{\mu}_{\phi_c}(\boldsymbol{x}_0^c)$ without considering variance [19]. After that, L-DiffRec concatenates $\{\boldsymbol{z}_0^c\}_{c=1}^C$ into z_0 for diffusion like DiffRec. Finally, by feeding the reconstructed $\hat{z}_0$ into the decoders, we will obtain $\hat{\boldsymbol{x}}_0$ for item ranking and generate top-K recommendations.

3.6 Temporal Diffusion

Since user preference might shift over time, it is crucial to capture temporal information during DiffRec learning. Assuming that more recent interactions can better represent users' current preferences, we propose a time-aware reweighting strategy to assign larger weights to users' later interactions.

Formally, for user u with M interacted items, the interaction time is available and the interaction sequence is formulated as $\mathcal{S} = \{i_1, i_2, \ldots, i_M\}$, where i_m denotes the ID of the m-th interacted item. We define the weights of interacted items $\boldsymbol{w} = [w_1, w_2, \ldots, w_M]$ via a time-aware linear schedule[2]: $w_m = w_{\min} + \frac{m-1}{M-1}(w_{\max} - w_{\min})$, where the two hyper-parameters $w_{\min} < w_{\max} \in (0, 1]$ represent the lower and upper bounds of interaction weights. Thereafter, the interaction history $\boldsymbol{x}_0$ of user u is reweighted as $\bar{\boldsymbol{x}}_0 = \boldsymbol{x}_0 \odot \bar{\boldsymbol{w}}$, where $\bar{\boldsymbol{w}} \in \mathbb{R}^{|\mathcal{I}|}$ is the weight vector calculated by $\boldsymbol{w}$, *i.e.,*

$$\bar{\boldsymbol{w}}[i] = \begin{cases} \boldsymbol{w}[\mathrm{Idx}(i)], & \text{if } i \in \mathcal{S} \\ 0, & \text{else} \end{cases} \tag{16}$$

where $\mathrm{Idx}(i)$ denotes the index of item i in the interaction sequence $\mathcal{S}$ of user u. By feeding the reweighted interaction history $\bar{\boldsymbol{x}}_0$ into DiffRec and L-DiffRec, we will obtain T-DiffRec and LT-DiffRec using temporal information, respectively.

[2]We use a linear schedule instead of the exponential scaling to simplify the reweighting strategy and save hyper-parameters, leaving more options to future work.

Table 1: Statistics of three datasets under two different settings, where "C" and "N" represent clean training and natural noise training, respectively. "Int." denotes interactions.

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

4 EXPERIMENTS

In this section, we conduct extensive experiments on three real-world datasets to answer the following research questions:

- **RQ1:** How does our DiffRec perform compared to the baselines under various experimental settings and how do the designs of DiffRec (*e.g.,* importance sampling, the inference step T', and the reduced noise scales) affect the performance?
- **RQ2:** How does L-DiffRec perform regarding the recommendation accuracy and resource costs?
- **RQ3:** Can T-DiffRec surpass sequential recommender models when interaction timestamps are available for training?

4.1 Experimental Settings

4.1.1 ***Datasets.*** We conduct experiments on three publicly available datasets in different scenarios. 1) **Amazon-book**[3] is from the Amazon review datasets, which covers rich user interactions with extensive books. 2) **Yelp**[4] is a representative business dataset containing user reviews for different restaurants. 3) **ML-1M**[5] is a popular benchmark dataset with user ratings on movies.

For all datasets, we first sort all interactions chronologically according to the timestamps. Thereafter, we consider three different training settings as follows. 1) **Clean training** discards user interactions with ratings < 4, and then splits the sorted interactions into training, validation, and testing sets with the ratio of 7:1:2. 2) **Noisy training** keeps the same testing set of clean training, but adds some noisy interactions, including natural noises (*i.e.,* the interactions with ratings < 4) and randomly sampled interactions into the training and validation sets. Note that we keep the numbers of noisy training and validation interactions on a similar scale as clean training for a fair comparison. 3) **Temporal training**: to evaluate the effectiveness of temporal modeling, we additionally consider using timestamps for training, *i.e.,* modeling the user interaction sequences like sequential recommender models. The testing set is also the same as clean and noisy training for a fair comparison. The dataset statistics are summarized in Table 1.

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

Table 2: Overall performance comparison between the baselines and DiffRec under clean training on three datasets. The best results are highlighted in bold and the second-best results are underlined. % Improve. represents the relative improvements of DiffRec over the best baseline results. $*$ implies the improvements over the best baseline are statistically significant (p-value < 0.05) under one-sample t-tests.

	Amazon-book				Yelp				ML-1M			
Methods	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**	**R@10**	**R@20**	**N@10**	**N@20**
MF	0.0437	0.0689	0.0264	0.0339	0.0341	0.0560	0.0210	0.0276	0.0876	0.1503	0.0749	0.0966
LightGCN	0.0534	0.0822	0.0325	0.0411	0.0540	0.0904	0.0325	0.0436	0.0987	0.1707	0.0833	0.1083
CDAE	0.0538	0.0737	0.0361	0.0422	0.0444	0.0703	0.0280	0.0360	0.0991	0.1705	0.0829	0.1078
MultiDAE	0.0571	0.0855	0.0357	0.0442	0.0522	0.0864	0.0316	0.0419	0.0995	0.1753	0.0803	0.1067
MultiDAE++	0.0580	0.0864	0.0363	0.0448	0.0544	0.0909	0.0328	0.0438	0.1009	0.1771	0.0815	0.1079
MultiVAE	0.0628	0.0935	0.0393	0.0485	0.0567	0.0945	0.0344	0.0458	0.1007	0.1726	0.0825	0.1076
CODIGEM	0.0300	0.0478	0.0192	0.0245	0.0470	0.0775	0.0292	0.0385	0.0972	0.1699	0.0837	0.1087
DiffRec	**0.0695***	**0.1010***	**0.0451***	**0.0547***	**0.0581***	**0.0960***	**0.0363***	**0.0478***	**0.1058***	**0.1787***	**0.0901***	**0.1148***
% Improve.	10.67%	8.02%	14.76%	12.78%	2.47%	1.59%	5.52%	4.37%	4.86%	0.90%	9.21%	6.69%

4.1.2 ***Baselines.*** We compare DiffRec with competitive baselines, including generative methods, and non-generative methods.

- **MF** [30] is one of the most representative collaborative filtering methods based on matrix factorization.
- **LightGCN** [7] learns user and item representations via the linear neighborhood aggregation on graph convolution networks.
- **CDAE** [46] trains an Auto-Encoder (AE) to recover the original user interactions from the randomly corrupted interactions.
- **MultiDAE** [19] uses dropout to corrupt the interactions and recover them via an AE with the multinomial likelihood.
- **MultiDAE++** is designed by us by adding noises to corrupt interactions similar to DiffRec and training a MultiDAE to recover clean interactions in a single decoding step. The added noises in MultiDAE++ are the same as DiffRec while DiffRec learns to denoise little by little in the reverse process.
- **MultiVAE** [19] utilizes VAEs to model the interaction generation process, where the posterior is approximated by an encoder.
- **CODIGEM** [36] is a generative model using the diffusion process, which adopts multiple AEs to model the reverse generation yet only utilizes the first AE for interaction prediction.

Evaluation. We follow the full-ranking protocol [7] by ranking all the non-interacted items for each user. For performance comparison, we adopt two widely used metrics Recall@K (R@K) and NDCG@K (N@K) over the top-K items, where K is set as 10 or 20.

4.1.3 ***Hyper-parameters Settings.*** We select the best hyper-parameters according to Recall@20 on the validation set. We tune the learning rates of all models in $\{1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}\}$. As to model-specific hyper-parameters, the search scopes are as follows.

- MF & LightGCN. The dropout ratio is selected from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. The weight decay is chosen from $\{1e^{-6}, 1e^{-5}, 1e^{-4}\}$ and the number of propagation layers is searched in $\{1, 2, 3\}$.

- CDAE & MultiDAE & MultiDAE++ & MultVAE. We tune the weight decay and dropout ratio in the scopes of $\{0, 1e^{-3}, 1e^{-1}\}$ and $\{0.1, 0.3, 0.5\}$, respectively. Besides, we choose the activation function of CDAE from {sigmoid, relu, tanh}. As to MultVAE, the regularization strength β and the annealing step are searched in $\{0, 0.3, 0.5, 0.7\}$ and $\{0, 200, 500\}$, respectively. The noises for MultiDAE++ are fixed consistently with DiffRec. The hidden size is set to the default value of $[200, 600]$.

- CODIGEM. The diffusion step is chosen from $\{2, 5, 10, 40, 50, 100\}$ and the noise β at each step is tuned in range of $\{5e^{-5}, 1e^{-4}, 5e^{-4}\}$. The hidden sizes of the multiple five-layer AEs are set to the default value of 200.

- DiffRec & L-DiffRec & T-DiffRec. The step embedding size is fixed at 10. We choose the hidden size of the MLP of $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ in $\{[300], [200, 600], [1000]\}$. The diffusion step T and the inference step T' are tuned in $\{2, 5, 10, 40, 50, 100\}$ and $\{0, \frac{T}{4}, \frac{T}{2}\}$, respectively. Besides, the noise scale s, the noise lower bound $\alpha_{\min}$, the noise upper bound $\alpha_{\max}$ are searched in $\{0, 1e^{-5}, 1e^{-4}, 5e^{-3}, 1e^{-2}, 1e^{-1}\}$, $\{5e^{-4}, 1e^{-3}, 5e^{-3}\}$, and $\{5e^{-3}, 1e^{-2}\}$, respectively. As to L-DiffRec, the dimension of $\boldsymbol{z}_0$ is set to 300 and the category number C is chosen from $\{1, 2, 3, 4, 5\}$. For T-DiffRec, $w_{\min}$ is tuned in $\{0.1, 0.3, 0.5\}$ and $w_{\max}$ is set to 1. More details are in our released code.

All experiments are done using a single Tesla-V100 GPU, except for ACVAE in Table 7 using A40 due to high computing costs.

4.2 Analysis of DiffRec (RQ1)

4.2.1 ***Clean Training.*** We first present the comparison between DiffRec and the baselines under clean training without using timestamps in Table 2, from which we have the following observations.

- Most generative methods (*i.e.,* MultiVAE, MultiDAE, MultiDAE++, CDAE) usually yield better performance than MF and LightGCN. These superior results are possibly attributed to the alignment between the generative modeling and the real-world interaction generation procedure. Among all generative methods, MultiVAE reveals impressive performance, especially on Amazon-book and Yelp. This is because it utilizes variational inference and multinomial likelihood [19], leading to stronger generation modeling.
- In all cases, our revised MultiDAE++ consistently outperforms MultiDAE. This implies the effectiveness of denoising training on enhancing the representation abilities of generative models. Besides, CODIGEM performs worse compared to LightGCN and other generative methods. This is fair because although multiple AEs are trained to model the forward and reverse processes, CODIGEM only uses the first AE for inference, and thus it is essentially learning a MultiDAE with the noises at a small scale.

Table 3: Performance comparison between DiffRec, the best generative baseline (MultiVAE), and the best non-generative baseline (LightGCN) under noisy training with natural noises.

	Amazon-book				Yelp				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
LightGCN	0.0400	0.0659	0.0231	0.0308	0.0466	0.0803	0.0278	0.0379	0.0648	0.1226	0.0470	0.0679
MultiVAE	0.0536	0.0820	0.0316	0.0401	0.0494	0.0834	0.0293	0.0396	0.0653	**0.1247**	0.0469	0.0680
DiffRec	**0.0546**	**0.0822**	**0.0335**	**0.0419**	**0.0507**	**0.0853**	**0.0309**	**0.0414**	**0.0658**	0.1236	**0.0488**	**0.0703**

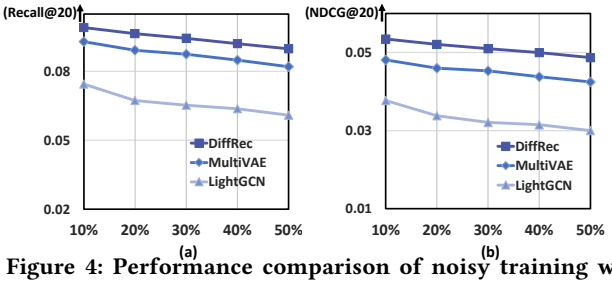

Figure 4: Performance comparison of noisy training with random noises on Amazon-book.

The inferior performance of CODIGEM than MultiVAE is also consistent with the results in Table 2 of [36].

- DiffRec significantly achieves superior performance on three datasets. The large improvements over VAE-based methods validate the superiority of applying DMs for recommender systems. Such improvements result from that 1) DiffRec is capable of modeling complex distributions via gradually learning each denoising transition step from t to $t-1$ with shared neural networks [31]; 2) DiffRec utilizes simple forward corruption for tractable posterior distribution, alleviating the intrinsic trade-off between the tractability and representation ability of VAE-based methods; and 3) notably, the scheduled noises for corruption in Eq. (4) ensure personalized preference modeling (*cf.* Section 3.4).

4.2.2 ***Noisy Training.*** In real-world recommender systems, collected user interactions in implicit feedback naturally contain false-positive and false-negative items. To analyze the performance of DiffRec on learning from noisy interactions, we compare DiffRec with the best non-generative method LightGCN and the best generative method MultiVAE under two noisy settings: 1) **natural noises**, where we randomly add some false-positive interactions with ratings < 4 as positive ones to the training and validation sets (see Section 4.1.1); and 2) **random noises**, where we randomly add a proportion of non-interacted items as positive interactions for each user. We summarize the performance of natural noises in Table 3 and the results of random noises with the noise proportion ranging from 10% to 50% in Figure 4. In Figure 4, we only show the results on Amazon-book to save space as we have similar observations on Yelp and ML-1M.

From Table 3, we can observe that DiffRec usually surpasses MultiVAE and LightGCN, verifying the strong robustness of DiffRec against natural noises. This is reasonable since such false-positive interactions are essentially corrupted interactions and DiffRec is intrinsically optimized to recover clean interactions iteratively from the corruption. By contrast, LightGCN is vulnerable to noisy interactions because it might amplify the negative effect of noises by emphasizing high-order propagation, thus leading to poor performance. In addition, the comparable results on ML-1M are because this dense dataset is relatively easier for prediction.

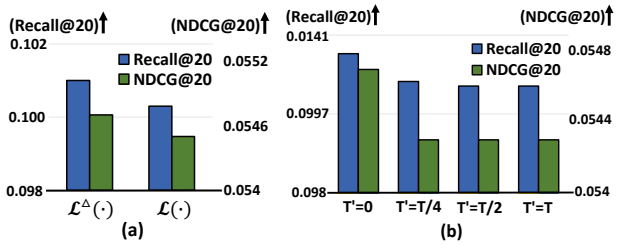

Figure 5: Effects of $\mathcal{L}^{\triangle}(\cdot)$, $\mathcal{L}(\cdot)$, and T', where $\mathcal{L}^{\triangle}(\cdot)$ and $\mathcal{L}(\cdot)$ mean importance sampling in Eq. (14) and uniform sampling in Eq. (13), respectively. T' is the inference step.

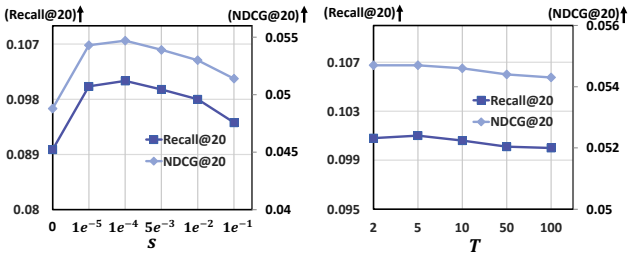

Figure 6: Effects of the noise scale s and diffusion step T.

From the results in Figure 4, we can find: 1) from adding 10% to 50% random noises, the performance of LightGCN, MultiVAE, and DiffRec gradually declines. This observation makes sense because it is harder to predict user preference as noises increase. Nevertheless, 2) DiffRec still outperforms MultiVAE and LightGCN even under a large scale of noises. The reason is that DiffRec is trained under different noise scales at each step, facilitating the recovery of real interactions from heavily corrupted interactions.

4.2.3 ***In-depth Analysis.*** We further explore the effects of different designs in DiffRec such as importance sampling, $\boldsymbol{x}_0$-ELBO, inference step T', and noise scales. The results on Amazon-book are reported in Figure 5 while the results on Yelp and ML-1M with similar observations are omitted to save space.

• **Importance sampling.** We compare the performance between importance sampling ($\mathcal{L}^{\triangle}(\cdot)$ in Eq. (14)) and uniform sampling ($\mathcal{L}(\cdot)$ in Eq. (13)) in Figure 5(a). The declined performance of $\mathcal{L}(\cdot)$ validates the effectiveness of importance sampling, which assigns large sampling probabilities to the large-loss steps and thus focuses on "hard" denoising steps for optimization.

• **Effect of inference step T'.** We vary T' from 0 to T during inference and show the results in Figure 5(b), from which we can find that using $T' = 0$ achieves better performance. It makes sense because collected interactions from real-world data naturally contain noises, and too much corruption might hurt personalization. In addition, the results are comparable when T' changes from $T/4$

Table 4: Performance comparison between L-DiffRec and DiffRec under natural noise training on three datasets.

	Amazon-book				Yelp				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
DiffRec	0.0546	0.0822	0.0335	0.0419	0.0507	0.0853	0.0309	0.0414	0.0658	0.1236	0.0488	0.0703
L-DiffRec	$0.0586^{+7.3\%}$	$0.0876^{+6.6\%}$	$0.0347^{+3.6\%}$	$0.0434^{+3.6\%}$	$0.0521^{+2.8\%}$	$0.0876^{+2.7\%}$	$0.0311^{+0.7\%}$	$0.0419^{+1.2\%}$	$0.0665^{+1.1\%}$	$0.1272^{+2.9\%}$	$0.0493^{+1.0\%}$	$0.0710^{+1.0\%}$

Table 5: Performance of ϵ-ELBO on ML-1M.

Variants	R@10	R@20	N@10	N@20
DiffRec (x_0-ELBO)	0.1058	0.1787	0.0901	0.1148
ϵ-ELBO	0.0157	0.0266	0.0170	0.0204

to T, possibly because the scheduled noise scale is relatively small, leading to minor changes in the ranking positions of top-K items.

• **Effects of noise scale s and step T.** In DiffRec, there are two important hyper-parameters: diffusion step T and noise scale s. To analyze their effect, we vary s at different scales and change T from 2 to 100, respectively. From the results in Figure 6, we can observe that: 1) as the noise scale increases, the performance first rises compared to training without noise ($s = 0$), verifying the effectiveness of denoising training. However, enlarging noise scales degrades the performance due to corrupting the personalization. Hence, we should carefully choose a relatively small noise scale (*e.g.,* $1e^{-4}$) as discussed in Section 3.4. And 2) the performance fluctuation *w.r.t.* T indicates that increasing diffusion steps has little effects on accuracy due to the relatively small noises in the forward process. Considering T being too large will cause high computing burdens, we choose $T = 5$ for good performance as well as low costs.

• **x_0-ELBO vs. ϵ-ELBO**. The comparison between predicting $\boldsymbol{x}_0$ and $\boldsymbol{\epsilon}$ ($\boldsymbol{\epsilon}$-ELBO, introduced in Section 3.4) on ML-1M is in Table 5. The results of $\boldsymbol{\epsilon}$-ELBO on Amazon-book and Yelp are close to zero due to severer data sparsity, and thus are omitted to save space. We attribute the worse results of $\boldsymbol{\epsilon}$-ELBO to the difficulty of predicting randomly sampled noises via an MLP. Besides, the reduced noise scales s may also enhance the prediction difficulty because the noises of different steps are becoming small with minor differences. We leave the further theoretical analysis to future work.

4.3 Analysis of L-DiffRec (RQ2)

To analyze L-DiffRec performance *w.r.t.* accuracy and resource costs, we evaluate L-DiffRec on three datasets under clean and noisy training. Moreover, we examine the effect of clustering category numbers to facilitate the future application of L-DiffRec.

4.3.1 ***Clean Training.*** From Table 6, we can find that L-DiffRec significantly outperforms MultiVAE with fewer resource costs (38.39% parameters and 10.61% GPU memory reduced on average), justifying the superiority of L-DiffRec. Meanwhile, it drastically lowers the costs of DiffRec with comparable accuracy, *i.e.,* reducing 56.17% parameters and 24.64% GPU usage on average. The comparable accuracy might be attributed to that the diffusion in the interaction space has redundant information and the dimension compression via clustering does not lose key information. The remarkable decline of resources is due to that 1) item clustering reduces the parameters of the encoder and decoder; 2) the latent diffusion lessens the parameters of the denoising MLP with θ. With significantly fewer resource costs, L-DiffRec has the potential to enable large-scale item prediction in industrial scenarios.

Table 6: Performance of L-DiffRec with $C = 2$, DiffRec, and MultiVAE under clean training. "par." denotes parameters.

Datasets	Method	R@10↑	R@20↑	N@10↑	N@20↑	#par.(M)↓	GPU(MB)↓
Amazon -book	**MultiVAE**	0.0628	0.0935	0.0393	0.0485	114	3,711
	DiffRec	**0.0695**	0.1010	**0.0451**	**0.0547**	190	5,049
	L-DiffRec	0.0694	**0.1028**	0.0440	0.0540	**75**	**3,077**
Yelp	**MultiVAE**	0.0567	0.0945	0.0344	0.0458	42	1,615
	DiffRec	0.0581	0.0960	**0.0363**	**0.0478**	69	2,103
	L-DiffRec	**0.0585**	**0.0970**	0.0353	0.0469	**29**	**1,429**
ML-1M	**MultiVAE**	0.1007	0.1726	0.0825	0.1076	4	497
	DiffRec	0.1058	0.1787	**0.0901**	**0.1148**	4	495
	L-DiffRec	**0.1060**	**0.1809**	0.0868	0.1122	**2**	**481**

4.3.2 ***Noisy Training.*** The resource costs of noisy training are the same as clean training while we observe that L-DiffRec consistently outperforms DiffRec under noisy training as shown in Table 4. One possible reason is that some clustered categories have few interactions, which are more likely to be false-positive interactions. The effect of such noises is weakened after representation compression via item clustering.

4.3.3 ***Effect of category number.*** To inspect the effect of category number on L-DiffRec, we compare the results with clustering category numbers changing from 1 to 5 on Amazon-book. We omitted similar results on Yelp and ML-1M to save space. From Figure 7, we can find that: 1) the Recall, NDCG, GPU usage, and parameters decline as the category number C increases as shown in Figure 7(a) and (b). This is reasonable since increasing C will reduce the parameters, hurting the representation ability. 2) The resource costs are substantially reduced compared to DiffRec and MultiVAE even if clustering is disabled ($C = 1$). This is due to the significant parameter reduction of the denoising MLP via latent diffusion. And 3) L-DiffRec is comparable with DiffRec when $C = 1$ or 2 while L-DiffRec consistently outperforms MultiVAE when $C = 1, 2$, or 3. As such, L-DiffRec can save extensive resources with comparable or superior accuracy by carefully choosing C.

4.4 Analysis of T-DiffRec (RQ3)

To verify the effectiveness of T-DiffRec on temporal modeling, we compare T-DiffRec and LT-DiffRec with a SOTA sequential recommender model ACVAE [47], which employs VAE with contrastive learning and adversarial training for recommendation.

From Table 7, we have the following observations: 1) T-DiffRec and LT-DiffRec perform better than DiffRec and L-DiffRec by a large margin, justifying the effectiveness of the proposed time-aware reweighting strategy on temporal modeling; 2) the superior performance of T-DiffRec and LT-DiffRec than ACVAE is attributed to both capturing temporal shifts and conducting diffusion processes, leading to more accurate and robust user representations; 3) despite more parameters, DiffRec-based methods consume much less GPU memory than ACVAE, thus reducing computing costs; 4) it is highlighted that LT-DiffRec yields comparable performance to T-DiffRec with fewer parameters, which is consistent with observations in Section 4.3; and 5) the relatively small improvements of T-DiffRec

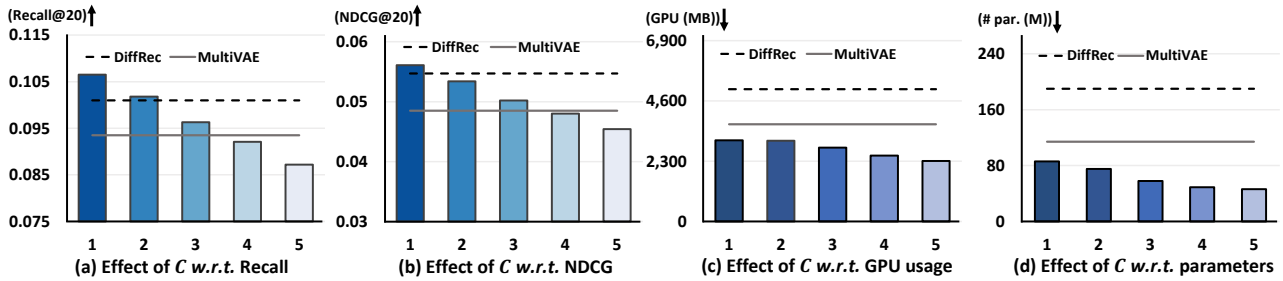

Figure 7: Effects of the clustering category number of L-DiffRec on Amazon-book under clean training.

Table 7: Performance comparison between DiffRec variants and a SOTA sequential baseline ACVAE. The models are trained using timestamps. The results on ML-1M are similar to Amazon-Book and omitted to save space. "par." denotes parameters.

	Amazon-book						Yelp					
	R@10↑	R@20↑	N@10↑	N@20↑	#par. (M)↓	GPU (MB)↓	R@10↑	R@20↑	N@10↑	N@20↑	#par. (M)↓	GPU (MB)↓
ACVAE	0.0770	0.1107	0.0547	0.0647	13	37,711	0.0567	0.0947	0.0342	0.0456	5	14,697
DiffRec	0.0695	0.1010	0.0451	0.0547	190	5,049	0.0581	0.0960	0.0363	0.0478	69	2,107
T-DiffRec	0.0819	0.1139	0.0565	0.0661	190	5,049	0.0601	0.0987	0.0377	0.0494	69	2,107
L-DiffRec	0.0694	0.1028	0.0440	0.0540	75	3,077	0.0585	0.0970	0.0353	0.0469	29	1,429
LT-DiffRec	0.0838	0.1188	0.0560	0.0665	75	3,077	0.0604	0.0982	0.0369	0.0484	29	1,429

over DiffRec on Yelp and the inferior results of ACVAE than DiffRec on Yelp are because user preference over food is relatively stable and the temporal shifts are limited. As such, considering temporal information receives minor benefits.

5 RELATED WORK

• **Generative recommendation.** Discriminative recommender models [20, 42] usually predict user-item interaction probabilities given the user and item representations. Although discriminative methods are cost-friendly, generative models can better learn collaborative signals between items due to simultaneously modeling the predictions over all items [28, 49]. Besides, generative models are specialized to capture the complex and non-linear relations between user preference and interactions as detailed in [16, 17, 33]. Existing generative recommender models can be roughly divided into two groups: GAN-based methods [6, 48] and VAE-based methods [21, 24]. GAN-based approaches utilize adversarial training [8, 37, 41, 45] to optimize the generator for predicting user interactions [3, 5, 13]. As to VAE-based methods [24, 50], they mainly learn an encoder for posterior estimation [9, 29], and a decoder to predict the interaction probabilities over all items [40]. For example, the most representative MultiVAE [19] achieves impressive performance by variational modeling.

Despite their success, DMs have shown great advantages over GANs and VAEs such as low instability and high generation quality in diverse tasks, including image synthesis [35], text generation [11], and audio generation [12]. As such, we consider revising DMs for generative recommendation.

• **Diffusion models.** DMs recently have shown the capability of high-quality generation [4, 26], covering conditional generation [2, 10, 22, 31] and unconditional generation [1, 15].

In spite of their success, utilizing DMs for recommendation receives little scrutiny. CODIGEM [36] claims to generate recommendation via DMs, which however is essentially a noise-based MultiDAE method [19] with inferior performance (*cf.* Table 2 in [36]). Specifically, CODIGEM iteratively introduces noises step by step and utilizes multiple different AEs for the prediction at each step. During inference, it estimates the interaction probabilities merely using the first AE, and thus the remaining AEs are totally useless. As such, CODIGEM differs from our DiffRec that employs a shared MLP for the multi-step prediction and considers the multi-step denoising for inference. In addition, some studies on social recommendation consider information diffusion on social networks [43, 44]. However, they mainly focus on the influence of social connections on user preference through diffusing processes [27], which intrinsically differ from DiffRec.

6 CONCLUSION AND FUTURE WORK

In this work, we proposed a novel DiffRec, which is a totally new recommender paradigm for generative recommender models. To ensure personalized recommendations, we reduced the noise scales and inference steps to corrupt users' interactions in the forward process. We also extended traditional DMs via two extensions to reduce the resource costs for large-scale item prediction and enable the temporal modeling of interaction sequences. Specifically, L-DiffRec clusters items for dimension compression and conducts diffusion processes in the latent space. Besides, T-DiffRec utilizes a time-aware reweighting strategy to capture the temporal patterns in users' interactions. Empirical results on three datasets under various settings validate the superiority of DiffRec with two extensions in terms of accuracy and resource costs.

This work opens up a new research direction for generative recommender models by employing DMs. Following this direction, many promising ideas deserve further exploration: 1) although L-DiffRec and T-DiffRec are simple yet effective, it is beneficial to devise better strategies to achieve better model compression and encode temporal information (*e.g.,* transformers); 2) it is meaningful to explore controllable or conditional recommendations based on DiffRec, *e.g.,* guiding the interaction prediction via a pre-trained classifier; and 3) exploring the effectiveness of more prior assumptions (*e.g.,* different noise assumptions other than Gaussian distribution) and diverse model structures is interesting.

REFERENCES

[1] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, Vol. 34. Curran Associates, Inc., 17981–17993.

[2] Chen-Hao Chao, Wei-Fang Sun, Bo-Wun Cheng, Yi-Chen Lo, Chia-Che Chang, Yu-Lun Liu, Yu-Lin Chang, Chia-Ping Chen, and Chun-Yi Lee. 2022. Denoising Likelihood Score Matching for Conditional Score-based Data Generation. *arXiv:2203.14206* (2022).

[3] Hao Chen, Zefan Wang, Feiran Huang, Xiao Huang, Yue Xu, Yishi Lin, Peng He, and Zhoujun Li. 2022. Generative Adversarial Framework for Cold-Start Item Recommendation. In *SIGIR*. ACM, 2565–2571.

[4] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. 2022. Diffusion models in vision: A survey. *arXiv:2209.04747* (2022).

[5] 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.* 546 (2021), 1166–1185.

[6] Guibing Guo, Huan Zhou, Bowei Chen, Zhirong Liu, Xiao Xu, Xu Chen, Zhenhua Dong, and Xiuqiang He. 2020. IPGAN: Generating informative item pairs by adversarial sampling. *TNNLS* 33, 2 (2020), 694–706.

[7] 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*. ACM, 639–648.

[8] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*. ACM, 355–364.

[9] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *NeurIPS*. Curran Associates, Inc., 6840–6851.

[11] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. In *NeurIPS*. Curran Associates, Inc., 12454–12465.

[12] Rongjie Huang, Zhou Zhao, Huadai Liu, Jinglin Liu, Chenye Cui, and Yi Ren. 2022. Prodiff: Progressive fast diffusion model for high-quality text-to-speech. In *MM*. ACM, 2595–2605.

[13] 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.

[14] 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.

[15] Max WY Lam, Jun Wang, Rongjie Huang, Dan Su, and Dong Yu. 2021. Bilateral denoising diffusion models. *arXiv:2108.11514* (2021).

[16] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *CIKM*. ACM, 811–820.

[17] Xiaopeng Li and James She. 2017. Collaborative Variational Autoencoder for Recommender Systems. In *KDD*. ACM, 305–314.

[18] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. In *arXiv:2205.14217*.

[19] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW*. ACM, 689–698.

[20] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. 2021. Interest-Aware Message-Passing GCN for Recommendation. In *WWW*. ACM, 1296–1305.

[21] Shuchang Liu, Fei Sun, Yingqiang Ge, Changhua Pei, and Yongfeng Zhang. 2021. Variation control and evaluation for generative slate recommendations. In *WWW*. ACM, 436–448.

[22] Xihui Liu, Dong Huk Park, Samaneh Azadi, Gong Zhang, Arman Chopikyan, Yuxiao Hu, Humphrey Shi, Anna Rohrbach, and Trevor Darrell. 2023. More control for free! image synthesis with semantic diffusion guidance. In *WACV*. IEEE, 289–299.

[23] Calvin Luo. 2022. Understanding diffusion models: A unified perspective. In *arXiv:2208.11970*.

[24] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning Disentangled Representations for Recommendation. In *NeurIPS*. Curran Associates, Inc., 5712–5723.

[25] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *ICML*. PMLR, 8162–8171.

[26] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. 2021. Grad-tts: A diffusion probabilistic model for text-to-speech. In *ICML*. PMLR, 8599–8608.

[27] Dimitrios Rafailidis and Fabio Crestani. 2017. Recommendation with social relationships via deep learning. In *SIGIR*. ACM, 151–158.

[28] Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding, and Ji-Rong Wen. 2020. Sequential recommendation with self-attentive multi-adversarial network. In *SIGIR*. ACM, 89–98.

[29] Zhaochun Ren, Zhi Tian, Dongdong Li, Pengjie Ren, Liu Yang, Xin Xin, Huasheng Liang, Maarten de Rijke, and Zhumin Chen. 2022. Variational Reasoning about User Preferences for Conversational Recommendation. In *SIGIR*. ACM, 165–175.

[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] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *CVPR*. IEEE, 10684–10695.

[32] Masahiro Sato, Sho Takemori, Janmajay Singh, and Tomoko Ohkuma. 2020. Unbiased Learning for the Causal Effect of Recommendation. In *RecSys*. ACM, 378–387.

[33] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I Nikolenko. 2020. Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. In *WSDM*. ACM, 528–536.

[34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*. PMLR, 2256–2265.

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. 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 minimax 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, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue. In *SIGIR*. ACM, 1288–1297.

[40] Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, Min Lin, and Tat-Seng Chua. 2022. Causal Representation Learning for Out-of-Distribution Recommendation. In *WWW*. ACM, 3562–3571.

[41] Zhidan Wang, Wenwen Ye, Xu Chen, Wenqiang Zhang, Zhenlei Wang, Lixin Zou, and Weidong Liu. 2022. Generative session-based recommendation. In *WWW*. ACM, 2227–2235.

[42] Yinwei Wei, Xiang Wang, Liqiang Nie, Shaoyu Li, Dingxian Wang, and Tat-Seng Chua. 2022. Causal Inference for Knowledge Graph based Recommendation. *TKDE* (2022).

[43] 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.

[44] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *SIGIR*. ACM, 235–244.

[45] Qiong Wu, Yong Liu, Chunyan Miao, Binqiang Zhao, Yin Zhao, and Lu Guan. 2019. PD-GAN: Adversarial Learning for Personalized Diversity-Promoting Recommendation.. In *IJCAI*, Vol. 19. 3870–3876.

[46] 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.

[47] Zhe Xie, Chengxuan Liu, Yichi Zhang, Hongtao Lu, Dong Wang, and Yue Ding. 2021. Adversarial and contrastive variational autoencoder for sequential recommendation. In *WWW*. ACM, 449–459.

[48] Lanling Xu, Jianxun Lian, Wayne Xin Zhao, Ming Gong, Linjun Shou, Daxin Jiang, Xing Xie, and Ji-Rong Wen. 2022. Negative Sampling for Contrastive Representation Learning: A Review. *arXiv:2206.00212* (2022).

[49] Xianwen Yu, Xiaoning Zhang, Yang Cao, and Min Xia. 2019. VAEGAN: A Collaborative Filtering Framework based on Adversarial Variational Autoencoders.. In *IJCAI*. AAAI Press, 4206–4212.

[50] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. Autosvd++ an efficient hybrid collaborative filtering model via contractive auto-encoders. In *SIGIR*. ACM, 957–960.