PREDIFF: Sequential Recommendation by Denoising Preference Distributions

Yaoqi Chen[1,2] **Jianjin Zhang**[1] **Qi Chen**[1] **Weihao Han**[1] **Yujing Wang**[1] **Hao Sun**[1]
[1]Microsoft [2]University of Science and Technology of China
yaoqi_chen@mail.ustc.edu.cn

Abstract

Sequential recommendation predicts which item a user will interact with next. A key property of this task is that user preferences are concentrated: only a small cluster of items is relevant at a given moment. Recent diffusion-based methods add noise to the item's embedding and score candidates by embedding similarity. Since similar items have similar embeddings, their score gaps are inherently small and easily overwhelmed by noise, causing the most relevant candidates to become indistinguishable first. We prove that moving diffusion to distribution space, where each item receives independent noise, preserves rankings better and makes the reverse denoising process easier. This shift, however, loses the semantic structure of embedding space and requires richer supervision than one-hot labels. We propose PREDIFF (**Pre**ference **Diff**usion), which performs diffusion in distribution space and introduces two components to address these problems: Distribution-Guided Sparse Projection projects the preference scores into embedding space while filtering out low-relevance noise, and soft targets encode inter-item similarity to teach fine-grained ranking within the preference cluster. On four benchmarks, PREDIFF achieves 8%–17% relative improvement over existing methods. The code is available at `https://github.com/Komorebi660/PreDiff`.

1 Introduction

Sequential recommendation aims to predict which item a user will engage with next based on their interaction history [Kang and McAuley, 2018, Sun et al., 2019, Hidasi et al., 2016]. A defining property of this task is that user preferences are concentrated: among thousands or millions of catalog items, only a handful are relevant at a given moment [Park and Tuzhilin, 2008, Steck, 2011, Abdollahpouri et al., 2019]. For example, after purchasing a phone case and charger, the set of plausible next items (e.g., cable organizers or screen protectors) is tiny relative to the full catalog.

Diffusion models [Ho et al., 2020, Sohl-Dickstein et al., 2015] gradually add noise to data and then learn to reverse this process. They have achieved strong results in image generation [Dhariwal and Nichol, 2021, Rombach et al., 2022] and have recently been applied to sequential recommendation [Li et al., 2024, Wang et al., 2023, Yang et al., 2023, Du et al., 2023, Lin et al., 2024, Li et al., 2025]. The appeal is natural: user histories are noisy and incomplete [Wang et al., 2021, Lu et al., 2018], and the iterative denoising process is well suited to recovering clean preference signals from such noisy input [Du et al., 2023, Yang et al., 2023]. However, image generation and recommendation have different needs. For images, many outputs are valid for the same prompt (e.g., a "cat photo" can vary in pose, color, and background), so spreading probability mass widely is helpful. For recommendation, as illustrated above, relevant items occupy only a small fraction of the catalog, so the model must concentrate its predictions rather than spread them across the full item catalog.

Current diffusion recommenders work in *embedding space* [Li et al., 2024, Yang et al., 2023, Chen et al., 2025]: they add noise to the target item's embedding and score all items by their inner product

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

with this noisy vector. Similar items have similar embeddings and small score gaps, so even low noise variance is enough to overwhelm these gaps and flip their ranking. We call this *noise coupling*: the most relevant candidates can become indistinguishable early in the forward process. This early loss of ranking information leaves the reverse process with weaker cues for recovering concentrated preferences, adding to its denoising burden. Existing methods attempt to mitigate this: DiffuRec [Li et al., 2024] reduces the noise magnitude and ADRec [Chen et al., 2025] adds multi-stage training, but noise coupling is a structural property of the space, not a training artifact.

Our key insight is to move diffusion from embedding space to *distribution space*. Instead of adding noise to the item embedding, we add noise to a preference vector that has one score per item. Because each item receives its own independent noise, similar items no longer interfere with each other. Rankings among top items are therefore preserved much longer during the forward process, making the reverse process easier. We prove this formally as the Ranking Preservation theorem (Theorem 1 in §3). However, exploiting these better-preserved ranking signals poses two challenges:

- **Lack of semantic structure.** Unlike embedding space where similar items are naturally close, distribution space carries no inter-item relationships. A natural fix is to project back by aggregating item embeddings weighted by preference scores, but since preferences are concentrated on a tiny fraction of items, the large number of low-relevance items collectively introduces noise into the aggregation and dilutes the signal from the few relevant ones.
- **Weak supervision.** A one-hot training label only indicates which single item is correct, but says nothing about its neighborhood. Without knowing which nearby items are also partially relevant, the model cannot learn the fine-grained ranking structure within the concentrated preference cluster.

We propose PREDIFF (**Pre**ference **Diff**usion), which performs diffusion in distribution space and introduces two components to address these challenges. To recover semantic structure, we design *Distribution-Guided Sparse Projection*, which samples a small subset of items proportional to the input preference scores and aggregates only their embeddings, filtering out dilution from the large number of low-relevance items. To provide richer supervision, we introduce *soft targets* that encode not only which item is correct but also which items are similar, teaching the model fine-grained ranking structure within the concentrated preference cluster during training. Experiments on four benchmarks show that PREDIFF achieves 8%–17% relative improvement over the strongest traditional and diffusion baselines.

2 Background and Related Work

2.1 Sequential Recommendation

Given a user's chronologically ordered interactions $i_1, i_2, \ldots, i_m$, sequential recommendation predicts the next item $i_\star$. The task requires modeling both item correlations [Kang and McAuley, 2018, Hidasi et al., 2016, Zhang et al., 2025] and evolving user interests [Lian et al., 2021, Chang et al., 2021]. Early methods used CNNs [Tang and Wang, 2018, Yuan et al., 2019] and RNNs [Hidasi and Karatzoglou, 2018, Hidasi et al., 2016]; Transformer-based models [Kang and McAuley, 2018, Sun et al., 2019] advanced the state of the art via self-attention over long histories. CL4SRec [Xie et al., 2022] further improved robustness through contrastive self-supervised learning. However, these methods are discriminative and output point estimates, which cannot express preference uncertainty [Park and Tuzhilin, 2008, Steck, 2011, Abdollahpouri et al., 2019]. VAE-based [Sachdeva et al., 2019, Xie et al., 2021] address this by modeling preferences as latent distributions, and GAN-based [Bharadhwaj et al., 2018, Ren et al., 2020] learn relevance distributions through adversarial training. Both capture preference uncertainty better than point estimates, but VAEs suffer from posterior collapse [Bowman et al., 2016, Liang et al., 2018] and GANs from training instability [Arjovsky et al., 2017]. Diffusion models offer a more stable generative alternative [Ho et al., 2020, Dhariwal and Nichol, 2021].

2.2 Diffusion Models

Diffusion models [Ho et al., 2020, Sohl-Dickstein et al., 2015] define a Markov chain of T forward steps that gradually corrupt data x_0 into Gaussian noise x_T, then train a network to reverse this process. In the forward process:

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

where $\{\beta_t\}_{t=1}^T$ controls the noise magnitude. Thanks to the Markov property, x_t can be computed directly from x_0:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$. The posterior $q(x_{t-1}|x_t, x_0)$ is also Gaussian:

$$q(x_{t-1}|x_t, x_0) \sim \mathcal{N}(c_{1,t} x_0 + c_{2,t} x_t,\ \tilde{\beta}_t \mathbf{I}), \tag{3}$$

with $c_{1,t} = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}$, $c_{2,t} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$, $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. We train a network f_θ to predict the clean input $\hat{x}_0 = f_\theta(x_t, t, c)$ from the noisy input x_t, timestep t, and conditional signal c. Then, the reverse step is:

$$p_\theta(x_{t-1}|x_t) \sim \mathcal{N}(c_{1,t}\hat{x}_0 + c_{2,t} x_t,\ \tilde{\beta}_t \mathbf{I}). \tag{4}$$

2.3 Diffusion for Sequential Recommendation

Diffusion's iterative denoising is a natural fit for recommendation, where interaction histories contain noise from misclicks and exploration [Lu et al., 2018, Wang et al., 2021]. Many diffusion-based methods have been proposed [Wang et al., 2023, Li et al., 2024, Yang et al., 2023, Lin et al., 2024, Du et al., 2023, Li et al., 2025, Huang et al., 2024, Chen et al., 2025, Cui et al., 2024].

Most of these methods operate in embedding space and face the noise coupling problem described in §1: similar items respond to noise almost identically, collapsing their rankings. DiffuRec [Li et al., 2024] mitigates this by using a very small blending weight to limit embedding drift; ADRec [Chen et al., 2025] uses three-stage training to stabilize embeddings; CaDiRec [Cui et al., 2024] adds contrastive augmentation to improve robustness. However, noise coupling is a structural property of embedding-space diffusion, so these strategies treat symptoms rather than the cause.

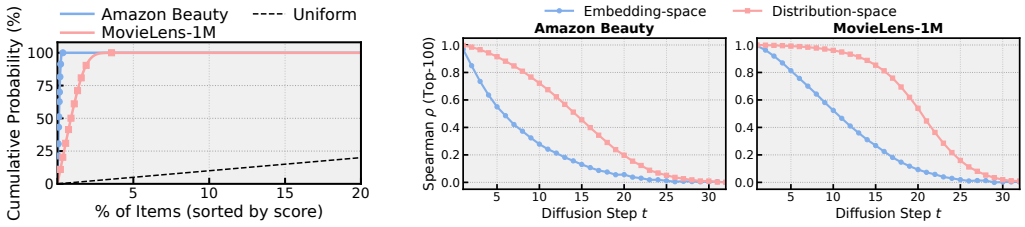
A complementary direction is to move beyond embedding-space. DiffRec [Wang et al., 2023] corrupts binary interaction vectors for matrix completion, while PreferGrow [Hu et al., 2025] models pairwise preference ratios through discrete fading and growing over the items. For reranking, DCDR [Lin et al., 2024] applies discrete diffusion to candidate lists, and DNR [Mao et al., 2026] denoises retriever scores over retrieved candidates using adversarial noise generation. In contrast, PREDIFF applies continuous Gaussian diffusion to a full-catalog preference vector for next-item prediction.

3 Why Distribution-Space Diffusion?

Before presenting our method, we provide empirical and theoretical evidence for why distribution-space diffusion is better suited to recommendation than the embedding-space approach.

User preferences are concentrated. Given a user's interaction history, most next-item probability mass typically concentrates on a small fraction of the items. We quantify this concentration on Amazon Beauty (N=12,101 items) [McAuley et al., 2015, He and McAuley, 2016] and MovieLens-1M (N=3,706 items) [Harper and Konstan, 2016]. We sample 10,000 windows from user histories and retrieve similar windows (Jaccard similarity > 0.5). Their next items are aggregated into an empirical distribution. As Figure 1(a) shows, preferences are highly concentrated: on Amazon Beauty, the top 0.2% of items cover 90% of cumulative probability; on MovieLens-1M, the top 1.9% suffice. Both curves plateau far above the uniform baseline, consistent with prior work on preference concentration [Park and Tuzhilin, 2008, Steck, 2011, Abdollahpouri et al., 2019]. This concentration does not imply that users have only one interest; multiple interest groups can still collectively occupy a small fraction of the catalog.

Noise coupling in embedding space. The goal of recommendation is to rank items correctly. In embedding-space diffusion, noise is added to the target item vector, and all item scores are inner products with this noisy vector. Similar items have small score gaps to begin with, so even a small amount of noise can flip their ranking. The most confusable items, which are the top-ranked candidates that matter most, are the first whose rankings get destroyed. Once these rankings are corrupted early in the forward process, the reverse process must recover concentrated preferences from a degraded ranking signal. We formalize this intuition by defining a pairwise ranking signal-to-noise ratio (SNR) and show that embedding-space diffusion introduces an additional source of ranking noise, beyond the noise inherent to the diffusion process itself.



(a) CDF of user preference

(b) Ranking preservation under different diffusion spaces

Figure 1: **(a) Cumulative next-item probability vs. fraction of items** (sorted by probability). Preferences are concentrated: less than 2% of items cover over 90% probability. **(b) Spearman ρ among top-100 items during DDPM forward process.** Distribution-space diffusion preserves rankings much longer, consistent with Theorem 1.

Definition 1 (Pairwise Ranking SNR)**.** *Let $s_k^{(t)}$ denote the relevance score of item k under the noised representation at diffusion step t. The **pairwise ranking SNR** between target item $i_\star$ and distractor item j is:*

$$\mathrm{SNR}_j(t) = \frac{\mathbb{E}[s_\star^{(t)} - s_j^{(t)}]}{\sqrt{\mathrm{Var}[s_\star^{(t)} - s_j^{(t)}]}}. \tag{5}$$

Higher SNR means the correct ranking is more likely preserved. We compare two approaches under standard DDPM, which differ in how $s_k^{(t)}$ is computed. **(A) Embedding-space diffusion**: noise is added to the target item embedding, $e_{\star,t} = \sqrt{\bar{\alpha}_t}\, e_\star + \sqrt{1-\bar{\alpha}_t}\, \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I_D)$, and scores are $s_k^{(t)} = e_k^\top e_{\star,t}$. **(B) Distribution-space diffusion**: noise is added to a preference vector, $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \eta$ with $\eta \sim \mathcal{N}(0, I_N)$, and scores are $s_k^{(t)} = (x_t)_k$. The following theorem compares their ranking SNR.

Theorem 1 (Ranking Preservation)**.** *Under DDPM with one-hot distribution-space target x_0 and embedding-space target $e_\star$, the pairwise ranking SNR between target item $i_\star$ and distractor item j with similarity score $c_{ji_\star} = e_j^\top e_\star$ is:*

$$\mathrm{SNR}_{\mathrm{emb}}(t) = \sqrt{\frac{\bar{\alpha}_t\,(1 - c_{ji_\star})}{2(1-\bar{\alpha}_t)}}, \qquad \mathrm{SNR}_{\mathrm{dist}}(t) = \sqrt{\frac{\bar{\alpha}_t}{2(1-\bar{\alpha}_t)}}. \tag{6}$$

Their ratio is independent of the noise schedule and item set size:

$$\frac{\mathrm{SNR}_{\mathrm{emb}}}{\mathrm{SNR}_{\mathrm{dist}}} = \sqrt{1 - c_{ji_\star}} \le 1, \quad \text{if } c_{ji_\star} \ge 0 \tag{7}$$

which holds for items most similar to the target. Full proof in Appendix A.

The degradation $\sqrt{1-c_{ji_\star}}$ is most severe for the most similar items, which are precisely the top-ranked candidates that matter most for recommendation quality. For example, when $c_{ji_\star} = 0.8$ (typical among top candidates), embedding-space diffusion retains only 45% of the distribution-space SNR for the same pair, meaning the ranking between the target and its closest competitors becomes highly unreliable. Figure 1(b) validates this on Amazon Beauty and MovieLens-1M. We randomly sampled 500 test data and measured Spearman's ρ[1] between the noised and clean rankings over top-100 items. Distribution-space diffusion maintains $\rho > 0.8$ for 10 steps, while embedding-space ρ collapses below 0.5 by step 10.

These results show that embedding-space diffusion introduces additional ranking noise beyond standard diffusion noise. By corrupting the relative order of top candidates early in the forward process, it weakens the ranking signal available during training, making it harder for the reverse process to learn how to recover the concentrated target preference distribution, potentially degrading

[1]Spearman's rank correlation coefficient measures the monotonic association between two rankings, defined as the Pearson correlation of rank variables [Spearman, 1904]. The embeddings used in this experiment come from trained PREDIFF.

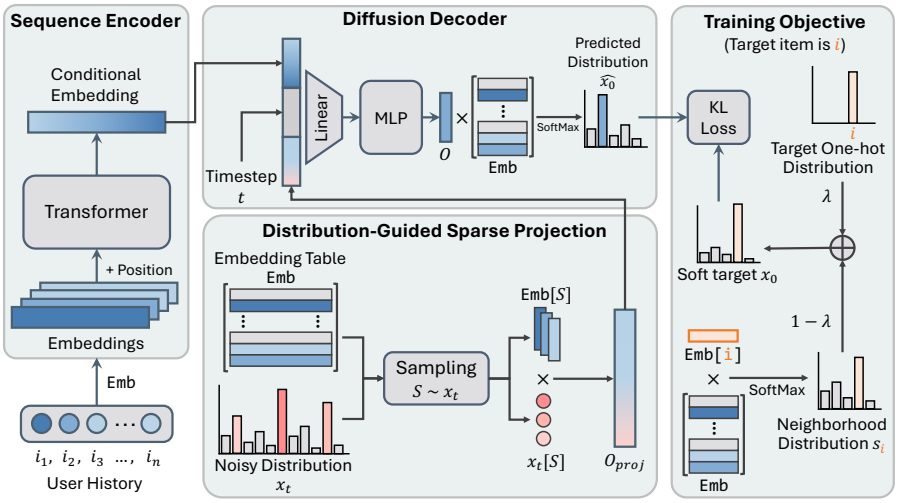


Figure 2: **Architecture of PREDIFF.** The Sequence Encoder encodes user history into a conditional embedding c. The Distribution-Guided Sparse Projection module samples a subset of items proportional to their noisy scores x_t and computes a weighted embedding O_{proj}, projecting N-dimensional distribution into D-dimensional space. The Diffusion Decoder then predicts the clean distribution $\hat{x}_0$, conditioned on user history c and timestep t. The model is trained end-to-end via soft targets.

recommendation quality. In contrast, distribution-space diffusion preserves the ranking structure far longer, providing a more informative learning signal for reverse recovery. Building on this insight, we propose PREDIFF in the next section.

4 Proposed Method: PREDIFF

Motivated by the analysis in §3, PREDIFF performs diffusion in distribution space, where the Markov states are per-item preference distributions $x_t \in \mathbb{R}^N$. Under the same DDPM noise schedule, Theorem 1 shows that distribution-space diffusion better preserves item rankings during the forward process than embedding-space diffusion, as noise propagates differently to item scores in the two spaces. However, this advantage alone does not guarantee effective reverse recovery: each dimension of distribution space represents an independent item score, without explicit geometric relationships between items (e.g., distances) to guide the denoiser in learning inter-item dependencies. To address this limitation, we introduce a *Distribution-Guided Sparse Projection* module to inject item relationships and *soft targets* to enrich the learning signal.

As shown in Figure 2, PREDIFF consists of three components: a Sequence Encoder that encodes user history, a Distribution-Guided Sparse Projection that bridges distribution space $\mathbb{R}^N$ and semantic space $\mathbb{R}^D$ while filtering out low-relevance noise, and a Diffusion Decoder that predicts the clean preference distribution.

4.1 Model Architecture

Sequence Encoder. The Sequence Encoder encodes user history into a conditional embedding c that guides denoising. Given the user's recent m items $i_1, \ldots, i_m$, we look up their D-dimensional embeddings from a shared table `Emb`, add learnable position embeddings, and process the sequence with a Transformer encoder [Vaswani et al., 2017], taking the last position's output as c:

$$c := f_\varphi(i_1, i_2, \cdots, i_m). \tag{8}$$

Distribution-Guided Sparse Projection. While diffusion runs in distribution space ($\mathbb{R}^N$), operating the decoder directly in this space faces three issues. First, distribution space carries no inter-item relationships: each dimension is just a score, so the decoder cannot exploit semantic similarity between items. Second, N can be very large (e.g., 59K for ML-25M), making an N-dimensional de-

coder computationally expensive. Third, the extreme sparsity of the target distribution (concentrated on a few items out of tens of thousands) makes learning in $\mathbb{R}^N$ harder than in a compact space.

A natural approach is to project back into embedding space ($\mathbb{R}^D$, $D \ll N$) by aggregating item embeddings weighted by their preference scores, i.e., $\mathrm{softmax}(x_t) \cdot \texttt{Emb}$. However, since preferences are concentrated on a tiny fraction of items, the large number of low-relevance items collectively introduces noise into the aggregation and dilutes the signal from the few relevant ones. We address this with *distribution dropout*: before aggregation, we stochastically drop a fraction d of items, retaining a subset $\mathcal{S}$ of size $\lfloor (1-d) \cdot N \rfloor$ sampled without replacement with probabilities proportional to $\mathrm{softmax}(x_t)$:

$$\mathcal{S} \sim \mathrm{Sample}\big(\lfloor (1-d) \cdot N \rfloor,\ \mathrm{softmax}(x_t)\big), \quad O_{\mathrm{proj}} = \mathrm{softmax}(x_t[\mathcal{S}]) \cdot \texttt{Emb}[\mathcal{S}]. \tag{9}$$

Since preferences are concentrated on a small number of items, most dropped items carry negligible probability. The sparse projection therefore loses little information while substantially reducing dilution from low-relevance items.

Like standard dropout [Srivastava et al., 2014], distribution dropout also prevents the decoder from overfitting to specific item combinations. The drop probabilities come from x_t itself, so the selection adapts to the noise level: at low noise, x_t concentrates near the target and the retained items naturally capture its nearest neighbors; at high noise, the retained items still preferentially include above-average candidates.

Diffusion Decoder. Since distribution-space diffusion preserves ranking structure longer (Theorem 1), the decoder faces less pressure to reconstruct fine-grained rankings at each step. This allows us to use a lightweight MLP instead of the Transformers used in embedding-space methods such as DiffuRec [Li et al., 2024] and ADRec [Chen et al., 2025], reducing inference latency. Specifically, the decoder concatenates three D-dimensional inputs (O_{proj} for the current preference state, a sinusoidal timestep embedding for the noise level, and c for user history) and processes them through L residual MLP blocks, producing $O \in \mathbb{R}^D$. This embedding is then projected back to item space via the shared embedding table $\texttt{Emb}$ and normalized with softmax to obtain the predicted preference distribution: $\hat{x}_0 = \mathrm{softmax}(\texttt{Emb} \cdot O)$. We denote the full decoder including the sparse projection process as:

$$\hat{x}_0 := f_\theta(x_t, t, c). \tag{10}$$

4.2 Model Training

A one-hot target treats all non-target items as equally wrong, but this ignores the structure of user preferences: a user who likes item i is more likely to also like items similar to i than completely unrelated ones. To encode this structure, we construct a soft target that blends the ground-truth item with its neighborhood:

$$x_0 = \lambda \cdot \mathrm{OneHot}(i) + (1-\lambda) \cdot \mathrm{softmax}(\mathbf{s}_i/\tau), \tag{11}$$

where $\mathbf{s}_i \in \mathbb{R}^N$ is the similarity score between target item i and all other items, λ controls how much probability mass stays on the target, and τ controls how sharply the neighborhood is defined. During training, the soft target provides a richer learning signal: instead of pushing all non-target items down equally, similar items receive less penalty, so the model learns to place them close in the predicted distribution. This directly addresses the weak supervision problem: a one-hot label says nothing about the neighborhood structure within the preference cluster, while the soft target explicitly encodes it. We provide gradient analysis and further discussion about soft targets in Appendix B.

We train the model to minimize the Kullback-Leibler (KL) divergence between the soft target and the predicted distribution:

$$\mathcal{L} = \mathbb{E}_{(x_0,c,t)} \left[\mathrm{KL}\left(x_0 \| f_\theta\left(x_t, t, c\right)\right)\right]. \tag{12}$$

This objective jointly trains the Encoder and Decoder. Algorithm 1 summarizes the procedure.

4.3 Model Inference

At inference, we start from pure noise $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively denoise using standard DDPM reverse sampling (Eq. 4):

$$x_{t-1} = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t}\, z, \quad z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{13}$$

After obtaining x_0, we exclude previously interacted items and recommend the top-scoring ones.

Algorithm 1 Training Phase

repeat
  Sample sequence $i_1, \cdots, i_m, i_\star$ from $\mathcal{D}$.
  $c = f_\varphi(i_1, \cdots, i_m)$ ▷ Encode history
  Sample $t \sim \text{Uniform}(\{1, \cdots, T\})$.
  $x_0 = \lambda \cdot \text{OneHot}(i_\star) + (1-\lambda) \cdot \text{softmax}(\mathbf{s}_{i_\star}/\tau)$
  Compute x_t by Eq.(2). ▷ Noise
  $\hat{x}_0 = f_\theta(x_t, t, c)$; compute $\mathcal{L}$ by Eq.(12). ▷ Diffusion denoise
  Update φ, θ via $\mathcal{L}$.
until converged

Algorithm 2 Inference Phase

Obtain sequence $i_1, \cdots, i_m$ from $\mathcal{D}_t$.
$c = f_\varphi(i_1, \cdots, i_m)$ ▷ Encode history
Sample $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
for $t = T, T-1, \cdots, 1$ **do**
  $\hat{x}_0 = f_\theta(x_t, t, c)$ ▷ Predict
  x_{t-1} by Eq.(13). ▷ Reverse step
end for
return x_0

5 Experiments

We evaluate PREDIFF on four questions: **(Q1)** How does PREDIFF perform compared to state-of-the-art sequential recommendation methods? **(Q2)** How does each component contribute to the overall performance? **(Q3)** How does PREDIFF's performance vary with key hyperparameters? **(Q4)** Does distribution-space diffusion better preserve target item ranking during reverse denoising?

5.1 Setup

Datasets We use four public benchmarks that cover a range of sparsity levels and item sizes. **Amazon Beauty** and **Toys and Games** [He and McAuley, 2016, McAuley et al., 2015] are sparse e-commerce datasets with short user histories. **MovieLens-1M** and **MovieLens-25M**[2] [Harper and Konstan, 2016] are denser movie-rating datasets, with ML-25M reaching 25 million interactions to stress-test scalability. Following prior work [Kang and McAuley, 2018, Sun et al., 2019, Li et al., 2024], we organize interactions chronologically and filter out users and items with fewer than 5 interactions. The maximum sequence length is 200 for MovieLens and 50 for Amazon datasets. We adopt leave-one-out evaluation: the last item for testing, the second-to-last for validation, and the rest for training. Dataset statistics are shown in Table 3 in the Appendix.

Baselines We compare against eight methods including traditional sequential recommenders and recent diffusion-based approaches. *Traditional:* **SASRec** [Kang and McAuley, 2018] uses a unidirectional Transformer to model sequences, **BERT4Rec** [Sun et al., 2019] adopts a bidirectional Transformer with masked item prediction, and **CL4SRec** [Xie et al., 2022] augments SASRec with contrastive learning. *Diffusion-based:* **DiffRec** [Wang et al., 2023] applies diffusion to interaction vectors, **DreamRec** [Yang et al., 2023] takes a purely generative diffusion approach in embedding space, **CaDiRec** [Cui et al., 2024] combines diffusion with contrastive augmentation, **DiffuRec** [Li et al., 2024] performs diffusion in the embedding space to model uncertainty, and **ADRec** [Chen et al., 2025] introduces token-level diffusion with three-stage training.

Implementation All experiments use PyTorch 2.0.1 on NVIDIA V100 GPUs. For PREDIFF, we fix a linear noise schedule with $T{=}32$ diffusion steps across all datasets. The sequence encoder uses 2 Transformer layers with $D{=}32$ on Amazon datasets, $D{=}128$ on ML-1M, and 4 layers with $D{=}128$ on ML-25M due to the larger dataset size; 4 MLP layers are used for the decoder throughout. We set $\lambda{=}0.8$ for the soft target and tune the τ over $\{0.05, 0.1, 0.2, 0.5, 1.0\}$ and distribution dropout rate over $\{0.3, 0.5, 0.7, 0.8, 0.9, 0.95\}$. All baselines follow their officially recommended hyperparameters, with model capacity scaled up for larger datasets. Detailed configurations are provided in Appendix D. For fair comparison, all methods fix the learning rate to 0.001 and batch size to 512 and train for up to 300K steps with early stopping (patience $= 10$). Following Li et al. [2024], we rank all non-interacted items per user and report Hit@K and NDCG@K for $K \in \{10, 20\}$.

5.2 Overall Performance (Q1)

Table 1 reports the recommendation results on four benchmarks. Among the baselines, diffusion-based methods generally outperform traditional sequential recommenders. DiffuRec and ADRec

[2]For brevity, we use ML-1M and ML-25M to denote MovieLens-1M and MovieLens-25M, respectively, hereafter.

Table 1: **Overall performance comparison (%).** All results are averaged over 3 random seeds with standard deviations reported. Best in **bold**, second-best underlined. Improvement is relative to the second-best. Latency is the average inference time per sample (ms) measured on NVIDIA V100 GPU. PREDIFF consistently achieves the best results across all datasets.

Dataset	Metric	BERT4Rec	SASRec	CL4SRec	DiffRec	DreamRec	CaDiRec	DiffuRec	ADRec	PREDIFF	Improv.
Beauty	Hit@10	$4.18_{\pm.14}$	$5.23_{\pm.02}$	$5.04_{\pm.07}$	$2.54_{\pm.21}$	$3.27_{\pm.31}$	$5.17_{\pm.11}$	$\underline{8.47}_{\pm.02}$	$6.16_{\pm.08}$	$\mathbf{9.44}_{\pm.13}$	11.45%
	Hit@20	$6.38_{\pm.25}$	$7.94_{\pm.05}$	$7.87_{\pm.08}$	$3.57_{\pm.22}$	$5.21_{\pm.28}$	$8.23_{\pm.12}$	$\underline{11.46}_{\pm.12}$	$8.47_{\pm.14}$	$\mathbf{12.88}_{\pm.26}$	12.39%
	NDCG@10	$2.18_{\pm.04}$	$2.79_{\pm.03}$	$2.55_{\pm.04}$	$1.41_{\pm.09}$	$1.79_{\pm.14}$	$2.60_{\pm.07}$	$\underline{5.05}_{\pm.03}$	$3.72_{\pm.06}$	$\mathbf{5.68}_{\pm.03}$	12.48%
	NDCG@20	$2.73_{\pm.04}$	$3.47_{\pm.02}$	$3.26_{\pm.04}$	$1.67_{\pm.09}$	$2.28_{\pm.13}$	$3.36_{\pm.08}$	$\underline{5.86}_{\pm.05}$	$4.29_{\pm.03}$	$\mathbf{6.55}_{\pm.05}$	11.77%
	Latency	*0.01*	*0.01*	*0.01*	*0.05*	*1.39*	*0.01*	*8.93*	*0.64*	*0.15*	–
Toys	Hit@10	$2.87_{\pm.10}$	$5.92_{\pm.29}$	$5.34_{\pm.05}$	$1.79_{\pm.17}$	$3.52_{\pm.26}$	$5.80_{\pm.20}$	$\underline{8.92}_{\pm.21}$	$6.16_{\pm.13}$	$\mathbf{10.21}_{\pm.04}$	14.46%
	Hit@20	$4.36_{\pm.21}$	$8.29_{\pm.35}$	$7.79_{\pm.06}$	$2.67_{\pm.14}$	$5.35_{\pm.20}$	$8.42_{\pm.14}$	$\underline{11.16}_{\pm.18}$	$8.43_{\pm.17}$	$\mathbf{13.02}_{\pm.04}$	16.67%
	NDCG@10	$1.50_{\pm.10}$	$3.58_{\pm.19}$	$2.98_{\pm.06}$	$0.98_{\pm.09}$	$1.98_{\pm.18}$	$3.19_{\pm.13}$	$\underline{5.61}_{\pm.15}$	$3.94_{\pm.05}$	$\mathbf{6.40}_{\pm.04}$	14.08%
	NDCG@20	$1.87_{\pm.12}$	$4.18_{\pm.20}$	$3.60_{\pm.04}$	$1.20_{\pm.08}$	$2.44_{\pm.16}$	$3.85_{\pm.12}$	$\underline{6.22}_{\pm.14}$	$4.54_{\pm.05}$	$\mathbf{7.18}_{\pm.02}$	15.43%
	Latency	*0.01*	*0.01*	*0.01*	*0.05*	*1.40*	*0.01*	*8.80*	*0.65*	*0.15*	–
ML-1M	Hit@10	$23.49_{\pm.61}$	$23.87_{\pm.19}$	$23.84_{\pm.74}$	$12.59_{\pm.23}$	$13.74_{\pm.53}$	$21.67_{\pm.62}$	$\underline{31.76}_{\pm.31}$	$26.81_{\pm.21}$	$\mathbf{34.74}_{\pm.17}$	9.38%
	Hit@20	$33.90_{\pm.73}$	$34.38_{\pm.30}$	$35.71_{\pm.63}$	$20.43_{\pm.17}$	$21.47_{\pm.74}$	$32.24_{\pm.78}$	$\underline{41.57}_{\pm.42}$	$37.69_{\pm.07}$	$\mathbf{45.77}_{\pm.23}$	10.10%
	NDCG@10	$12.45_{\pm.47}$	$12.86_{\pm.12}$	$12.72_{\pm.37}$	$6.38_{\pm.13}$	$7.23_{\pm.20}$	$11.56_{\pm.19}$	$\underline{18.47}_{\pm.39}$	$15.10_{\pm.04}$	$\mathbf{20.43}_{\pm.14}$	10.61%
	NDCG@20	$15.07_{\pm.49}$	$15.50_{\pm.09}$	$15.71_{\pm.34}$	$8.35_{\pm.14}$	$9.17_{\pm.25}$	$14.23_{\pm.19}$	$\underline{21.15}_{\pm.48}$	$17.66_{\pm.10}$	$\mathbf{23.20}_{\pm.08}$	9.69%
	Latency	*0.09*	*0.09*	*0.07*	*0.09*	*1.59*	*0.07*	*31.81*	*3.55*	*0.20*	–
ML-25M	Hit@10	$19.92_{\pm.26}$	$14.24_{\pm.07}$	$15.67_{\pm.52}$	$9.53_{\pm.07}$	$9.99_{\pm.42}$	$9.01_{\pm.22}$	$21.34_{\pm.03}$	$\underline{21.60}_{\pm.05}$	$\mathbf{23.71}_{\pm.27}$	9.77%
	Hit@20	$28.50_{\pm.21}$	$22.72_{\pm.10}$	$23.97_{\pm.70}$	$14.97_{\pm.06}$	$14.76_{\pm.72}$	$14.58_{\pm.48}$	$29.65_{\pm.05}$	$\underline{29.98}_{\pm.16}$	$\mathbf{32.83}_{\pm.28}$	9.51%
	NDCG@10	$11.00_{\pm.14}$	$7.13_{\pm.07}$	$8.20_{\pm.29}$	$5.00_{\pm.05}$	$5.49_{\pm.18}$	$4.62_{\pm.09}$	$12.54_{\pm.08}$	$\underline{12.60}_{\pm.03}$	$\mathbf{13.67}_{\pm.15}$	8.49%
	NDCG@20	$13.16_{\pm.13}$	$9.26_{\pm.08}$	$10.29_{\pm.34}$	$6.37_{\pm.05}$	$6.69_{\pm.25}$	$6.02_{\pm.15}$	$14.63_{\pm.07}$	$\underline{14.71}_{\pm.10}$	$\mathbf{15.97}_{\pm.13}$	8.57%
	Latency	*0.10*	*0.18*	*0.14*	*0.19*	*1.80*	*0.14*	*42.17*	*5.69*	*0.41*	–

achieve the best or second-best baseline results on all four datasets. This confirms that iterative denoising is effective for recovering user preferences from noisy interaction histories.

PREDIFF achieves the best performance across all datasets, improving over the strongest baseline by 8%–17%. The gains are largest on the two Amazon datasets (11.45%–16.67%), as both are sparse datasets where embedding-space diffusion must recover concentrated preferences from heavily corrupted rankings. In this regime, noise coupling makes the reverse process harder. By moving diffusion to distribution space, PREDIFF sidesteps this problem. On MovieLens datasets, the improvement narrows to $\sim$10%, because these datasets are much denser and larger (as shown in Table 3), so each item receives far more training interactions and the resulting embeddings are better separated, partially mitigating the noise coupling problem. Beyond accuracy, PREDIFF is also 4–$100\times$ faster than DiffuRec and ADRec at inference under the same $T{=}32$ diffusion steps, because the lightweight MLP denoiser reduces per-step cost compared to their Transformer decoders.

5.3 Ablation Study (Q2)

We evaluate each component of PREDIFF by removing it in isolation and measuring the performance drop relative to the full model. Table 2 shows the results.

Removing distribution-space diffusion from PREDIFF leads to a notable performance drop, bringing its results closer to those of a Transformer Encoder trained with full-category softmax cross-entropy (CE) and soft targets (e.g., Table 6 in Appendix E.3). This suggests that distribution-space diffusion is a key contributor to PREDIFF's effectiveness, while the training techniques such as full-category softmax CE and soft targets provide additional gains.

Table 2: **Ablation study (%).** All components contribute positively.

Settings	Beauty		ML-1M	
	Hit@20	NDCG@20	Hit@20	NDCG@20
w/o dist. diffusion	$8.37_{\pm.13}$	$4.24_{\pm.05}$	$37.68_{\pm.39}$	$17.60_{\pm.13}$
w/o sparse proj.	$1.86_{\pm.08}$	$0.73_{\pm.05}$	$6.75_{\pm.47}$	$2.49_{\pm.18}$
w/o soft target	$12.09_{\pm.05}$	$6.06_{\pm.02}$	$42.87_{\pm.31}$	$20.79_{\pm.35}$
Full PREDIFF	$\mathbf{12.88}_{\pm.26}$	$\mathbf{6.55}_{\pm.05}$	$\mathbf{45.77}_{\pm.23}$	$\mathbf{23.20}_{\pm.08}$

The sparse projection is the most critical component. Although distribution-space diffusion better preserves ranking signals, this alone does not guarantee that the diffusion decoder can leverage them to recover user preferences more effectively. Without sparse projection, the decoder operates directly on raw N-dimensional preference scores that lack explicit semantic relationships between items, leading to significant performance drops on both datasets. This is consistent with the poor performance of DiffRec in Table 1, which also operates directly on N-

dimensional interaction vectors. Projecting preference scores back into embedding space is therefore essential for recovering inter-item semantic structure.

Soft targets contribute 6%–11% relative improvement. Switching to pure one-hot supervision removes the neighborhood signal and treats all non-target items as equally irrelevant, leading to consistent drops on both datasets. PREDIFF still slightly outperforms the best embedding-space diffusion baseline DiffuRec even without soft targets (e.g., 42.87% vs. 41.57% on Movielens-1M Hit@20), confirming that the distribution-space design alone provides a strong foundation on which soft targets build further gains.

5.4 Hyperparameter Sensitivity (Q3)

We evaluate the sensitivity of PREDIFF to its key hyperparameters: the soft target coefficient λ, the soft target temperature τ, and the distribution dropout rate d. Figure 3 shows Hit@20 results on Amazon Beauty and Movielens-1M datasets. Additional results are in Appendix E.1.

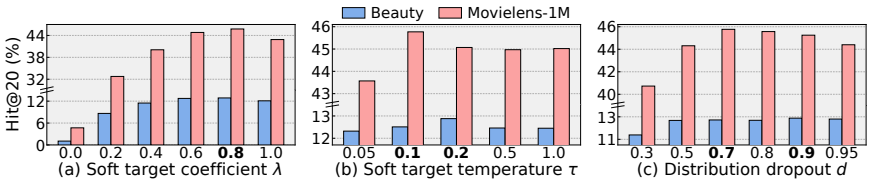


Figure 3: **Hyperparameter sensitivity.** (a) Soft target coefficient λ. (b) Soft target softmax temperature τ. (c) Distribution dropout rate d. Best values are shown in **bold**.

The soft target coefficient λ interpolates between one-hot and soft supervision. Without the ground-truth anchor (λ=0), the model cannot identify the correct item and performance collapses on both datasets. Performance improves steadily as λ increases toward 0.8, where the one-hot component anchors the target while the soft component provides neighborhood information. The performance drop at λ=1.0 confirms that the neighborhood signal helps.

The temperature τ controls how sharply the soft target concentrates probability on the most similar items. Too small a τ makes the soft target nearly one-hot and loses the neighborhood signal, while too large a τ spreads probability too uniformly and dilutes it. According to the experiments, the best results fall in $\tau \in [0.1, 0.2]$.

The distribution dropout rate d controls how many items are dropped in the sparse projection. Performance improves as d increases from 0.3, because dropping more low-relevance items reduces signal dilution in the aggregation. The optimum is d=0.9 for Amazon Beauty and d=0.7 for Movielens-1M, because Beauty has a much larger item set relative to the number of training interactions (Table 3) and therefore requires more aggressive filtering.

5.5 Ranking Preservation Analysis (Q4)

To directly verify whether distribution-space diffusion better preserves the ranking of the target item during denoising, we randomly sample 2000 test users and track the average rank of the ground-truth target item at each reverse diffusion step. We compare PREDIFF against two embedding-based diffusion recommenders, DiffuRec and ADRec. The results are shown in Figure 4.

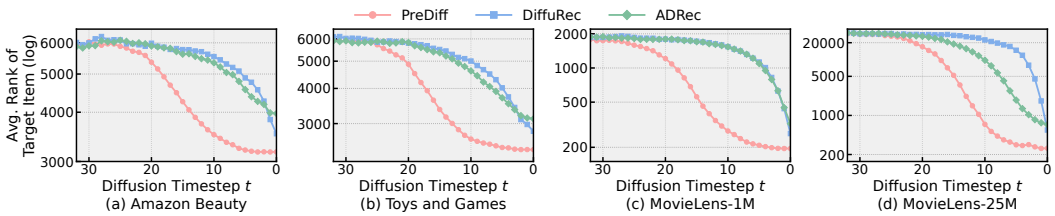


Figure 4: **Target item ranking during reverse diffusion** (Lower ranking is better). PREDIFF recovers the target item ranking faster and reaches a better final rank on all four datasets.

The average ranking of target items starts near $N/2$ at step T, indicating that the initial state of the reverse diffusion process is effectively random with no ranking information. As denoising proceeds, PREDIFF consistently recovers the target item ranking faster than both DiffuRec and ADRec. At the final step t=0, PREDIFF achieves 10%–53% lower rank than DiffuRec and 20%–62% lower than ADRec across all four datasets. This trend is consistent with the ranking preservation guarantee in Theorem 1 and confirms that operating in distribution space allows each denoising step to more effectively recover the target item ranking.

6 Conclusion

We showed that embedding-space diffusion suffers from noise coupling in sequential recommendation, where noise on the target vector erases the score gaps between similar items, making the most relevant candidates hardest to distinguish. Moving diffusion to distribution space gives each item independent noise and preserves rankings better, as we proved formally in the Ranking Preservation theorem. To make distribution-space diffusion practical, PREDIFF introduces Distribution-Guided Sparse Projection, which samples items proportional to their preference scores to recover semantic structure while filtering out low-relevance noise, and soft targets that encode inter-item similarity to provide richer supervision beyond one-hot labels. Experiments on four benchmarks show 8%–17% improvement over the baselines. We hope this work encourages further exploration of alternative diffusion spaces beyond embeddings for recommendation tasks.

Limitations and Future Work

Although PREDIFF is significantly faster than prior diffusion-based recommenders, it still requires T-step iterative sampling at inference, which is slower than single-pass methods like SASRec. Recent advances in one-step generation, such as flow matching [Lipman et al., 2023] and consistency models [Song et al., 2023], offer promising directions to reduce this latency while retaining the benefits of diffusion models. Additionally, our soft targets rely on embedding similarity to define inter-item neighborhoods. For items with very few interactions, the learned embeddings may not accurately reflect true similarity, potentially limiting the quality of the soft supervision signal. Incorporating item side information (e.g., text descriptions or category attributes) could address this and is a natural direction for future work.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers and the area chair for their valuable feedback. Microsoft provided the resources and support for this work. The authors declare no other competing interests.

References

Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. *arXiv preprint arXiv:1907.13286*, 2019.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223, 2017.

Homanga Bharadhwaj, Homin Park, and Brian Y. Lim. RecGAN: Recurrent generative adversarial networks for recommendation systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 372–376, 2018.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of the 20th Conference on Computational Natural Language Learning*, pages 10–21, 2016.

Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Sequential recommendation with graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 378–387, 2021.

Jialei Chen, Yuanbo Xu, and Yiheng Jiang. Unlocking the power of diffusion models in sequential recommendation: A simple and effective approach. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 155–166, 2025.

Ziqiang Cui, Haolun Wu, Bowei He, Ji Cheng, and Chen Ma. Context matters: Enhancing sequential recommendation with context-aware diffusion-based contrastive learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 404–414, 2024.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems 34*, pages 8780–8794, 2021.

Hanwen Du, Huanhuan Yuan, Zhen Huang, Pengpeng Zhao, and Xiaofang Zhou. Sequential recommendation with diffusion models. *arXiv preprint arXiv:2304.04541*, 2023.

F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2016.

Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517, 2016.

Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 843–852, 2018.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33*, 2020.

Guoqing Hu, An Zhang, Shuchang Liu, Wenyu Mao, Jiancan Wu, Xun Yang, Xiang Li, Lantao Hu, Han Li, Kun Gai, and Xiang Wang. Fading to grow: Growing preference ratios via preference fading discrete diffusion for recommendation. In *Advances in Neural Information Processing Systems*, volume 38, pages 135766–135804, 2025.

Hongtao Huang, Chengkai Huang, Xiaojun Chang, Wen Hu, and Lina Yao. Dual conditional diffusion models for sequential recommendation. *arXiv preprint arXiv:2410.21967*, 2024.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining*, pages 197–206, 2018.

Wuchao Li, Rui Huang, Haijun Zhao, Chi Liu, Kai Zheng, Qi Liu, Na Mou, Guorui Zhou, Defu Lian, Yang Song, Wentian Bao, Enyun Yu, and Wenwu Ou. DimeRec: A unified framework for enhanced sequential recommendation via generative diffusion models. In *Proceedings of the 18th ACM International Conference on Web Search and Data Mining*, pages 726–734, 2025.

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

Jianxun Lian, Iyad Batal, Zheng Liu, Akshay Soni, Eun Yong Kang, Yajun Wang, and Xing Xie. Multi-interest-aware user modeling for large-scale sequential recommendations. *arXiv preprint arXiv:2102.09211*, 2021.

Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the World Wide Web Conference*, pages 689–698, 2018.

Xiao Lin, Xiaokai Chen, Chenyang Wang, Hantao Shu, Linfeng Song, Biao Li, and Peng Jiang. Discrete conditional diffusion for reranking in recommendation. In *Companion Proceedings of the ACM Web Conference*, pages 161–169, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The 11th International Conference on Learning Representations*, 2023.

Hongyu Lu, Min Zhang, and Shaoping Ma. Between clicks and satisfaction: Study on multi-phase user preferences and satisfaction for online news reading. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 435–444, 2018.

Wenyu Mao, Shuchang Liu, Hailan Yang, Xiaobei Wang, Xiaoyu Yang, Xu Gao, Xiang Li, Lantao Hu, Han Li, Kun Gai, An Zhang, and Xiang Wang. Denoising neural reranker for recommender systems. In *International Conference on Learning Representations*, volume 2026, pages 129105–129128, 2026.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52, 2015.

Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2nd ACM Conference on Recommender Systems*, pages 11–18, 2008.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, Maciej Kula, Ed Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Advances in Neural Information Processing Systems*, volume 36, pages 10299–10315, 2023.

Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding, and Ji-Rong Wen. Sequential recommendation with self-attentive multi-adversarial network. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 89–98, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10674–10685, 2022.

Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, pages 600–608, 2019.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265, 2015.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252, 2023.

Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the 5th ACM Conference on Recommender Systems*, pages 125–132, 2011.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1441–1450, 2019.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 565–573, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008, 2017.

Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Denoising implicit feedback for recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 373–381, 2021.

Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 832–841, 2023.

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. In *Proceedings of the 38th IEEE International Conference on Data Engineering*, pages 1259–1273, 2022.

Zhe Xie, Chengxuan Liu, Yichi Zhang, Hongtao Lu, Dong Wang, and Yue Ding. Adversarial and contrastive variational autoencoder for sequential recommendation. In *Proceedings of the Web Conference*, pages 449–459, 2021.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. In *Advances in Neural Information Processing Systems 36*, 2023.

Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, pages 582–590, 2019.

Yanlin Zhang, Yuchen Shi, Deqing Yang, and Xiaodong Gu. Exploiting explicit item-item correlations from knowledge graphs for enhanced sequential recommendation. *Information Systems*, 128: 102470, 2025.

A Proof of Theorem 1 (Ranking Preservation under Diffusion)

We observe that learned item embeddings in PREDIFF have near-uniform norms in practice, for example, on the MovieLens-1M dataset, the variation of $\|e_i\|$ is only 10.0%. To simplify the analysis, we assume unit-norm embeddings ($\|e_i\| = 1$), so that $c_{ji_\star} = e_j^\top e_\star$ reduces to the cosine similarity under this assumption. This assumption does not affect ranking: rescaling all embeddings by a global constant preserves dot-product orderings.

Let $E = [e_1, \ldots, e_N]^\top \in \mathbb{R}^{N \times D}$ be the item embedding matrix with unit-norm rows, and $c_{ji_\star} = e_j^\top e_\star$ the inner product between distractor j and target $i_\star$. Consider a target user whose ground-truth next item is $i_\star$. In distribution space, this is represented as a one-hot vector $x_0 = \mathrm{OneHot}(i_\star) \in \mathbb{R}^N$ with $(x_0)_\star = 1$ and $(x_0)_k = 0$ for $k \neq \star$. In embedding space, the target representation is $e_\star \in \mathbb{R}^D$.

For a distractor item $j \neq i_\star$, we define the pairwise score difference at diffusion step t as $\Delta_j^{(t)} := s_\star^{(t)} - s_j^{(t)}$, where $s_j^{(t)}$ is the relevance score of item j under the noised representation. A positive $\Delta_j^{(t)}$ means the target item $i_\star$ is correctly ranked above distractor j. The pairwise ranking SNR is defined as:

$$\mathrm{SNR}_j(t) = \frac{\mathbb{E}[\Delta_j^{(t)}]}{\sqrt{\mathrm{Var}[\Delta_j^{(t)}]}}. \tag{14}$$

A higher SNR means the expected score gap is large relative to its fluctuation, i.e., the correct ranking is more robust to noise. We now prove the Theorem 1.

Embedding-Space Diffusion (Approach A). Under the standard DDPM forward process in embedding space, the target item embedding $e_\star$ is corrupted by isotropic Gaussian noise:

$$e_{\star,t} = \sqrt{\bar{\alpha}_t}\, e_\star + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I_D). \tag{15}$$

The relevance score for item k is computed as the inner product $s_k^A = e_k^\top e_{\star,t}$. Since all item scores are computed via inner products with the same noisy vector $e_{\star,t}$, similar items whose embeddings point in similar directions respond to the noise in nearly the same way, which is the source of noise coupling. The pairwise score difference between target $i_\star$ and distractor j is:

$$\Delta_j^A = e_\star^\top e_{\star,t} - e_j^\top e_{\star,t} = (e_\star - e_j)^\top e_{\star,t} = \underbrace{\sqrt{\bar{\alpha}_t}(e_\star - e_j)^\top e_\star}_{\text{signal}} + \underbrace{\sqrt{1 - \bar{\alpha}_t}(e_\star - e_j)^\top \varepsilon}_{\text{noise}}. \tag{16}$$

Taking the expectation, the signal is $\mathbb{E}[\Delta_j^A] = \sqrt{\bar{\alpha}_t}(1 - c_{ji_\star})$, which is proportional to $(1 - c_{ji_\star})$, so more similar items j and $i_\star$ produce a smaller expected score gap. The noise term $(e_\star - e_j)^\top \varepsilon$ is a linear combination of independent Gaussian variables whose variance is $\mathrm{Var}[\Delta_j^A] = (1 - \bar{\alpha}_t)\|e_\star - e_j\|^2 = 2(1 - \bar{\alpha}_t)(1 - c_{ji_\star})$, also proportional to $(1 - c_{ji_\star})$. Combining these, the pairwise ranking SNR becomes:

$$\boxed{\mathrm{SNR}_{\mathrm{emb}}(t) = \frac{\sqrt{\bar{\alpha}_t}(1 - c_{ji_\star})}{\sqrt{2(1 - \bar{\alpha}_t)(1 - c_{ji_\star})}} = \sqrt{\frac{\bar{\alpha}_t\,(1 - c_{ji_\star})}{2(1 - \bar{\alpha}_t)}}.} \tag{17}$$

As $c_{ji_\star} \to 1$ (highly similar items), the SNR vanishes, meaning that even a small amount of noise is enough to flip the ranking between the target and a similar distractor.

Distribution-Space Diffusion (Approach B). Under the DDPM forward process in distribution space, noise is added to the preference vector:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \eta, \quad \eta \sim \mathcal{N}(0, I_N). \tag{18}$$

Here η is an N-dimensional vector whose components η_k are independent standard Gaussians. The relevance score for item k is simply the k-th component $s_k^B = (x_t)_k$, and since each item's noise is independent, the noise affecting different items' scores is uncorrelated, unlike in embedding-space diffusion. Since $x_0 = \mathrm{OneHot}(i_\star)$, the pairwise score difference is:

$$\Delta_j^B = (x_t)_\star - (x_t)_j = \underbrace{\sqrt{\bar{\alpha}_t} \cdot 1}_{\text{signal}} + \underbrace{\sqrt{1 - \bar{\alpha}_t}\,(\eta_\star - \eta_j)}_{\text{noise}}. \tag{19}$$

The signal $\mathbb{E}[\Delta_j^B] = \sqrt{\bar{\alpha}_t}$ is now a constant independent of $c_{ji_\star}$, meaning the target always has a fixed score advantage regardless of how similar j and $i_\star$ are in embedding space. Because $\eta_\star$ and η_j are independent standard Gaussians, the noise variance $\mathrm{Var}[\Delta_j^B] = 2(1-\bar{\alpha}_t)$ is also constant, yielding an SNR of:

$$\boxed{\mathrm{SNR}_{\mathrm{dist}}(t) = \frac{\sqrt{\bar{\alpha}_t}}{\sqrt{2(1-\bar{\alpha}_t)}}.} \tag{20}$$

Unlike embedding-space diffusion, the distribution-space SNR does not depend on $c_{ji_\star}$. Because each item receives independent noise, the inter-item similarity structure in embedding space cannot amplify confusion. □

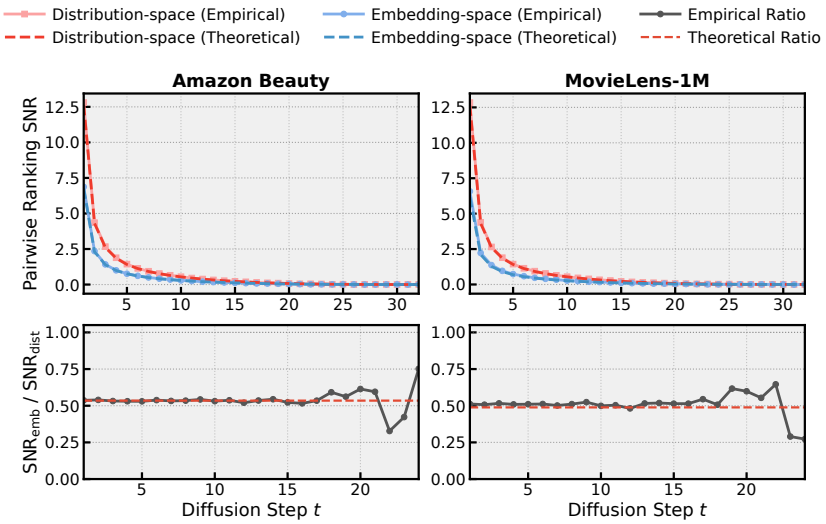


Figure 5: **Empirical verification of Theorem 1.** Top row: pairwise ranking SNR at each diffusion step for embedding-space (blue) and distribution-space (red) diffusion. Dashed lines show theoretical predictions while solid lines show the empirical SNR measured from 50 noise samples. Bottom row: empirical SNR ratio $\mathrm{SNR}_{\mathrm{emb}}/\mathrm{SNR}_{\mathrm{dist}}$ across steps (gray solid) vs. the theoretical value $\sqrt{1-c_{ji_\star}}$ (orange dashed). The ratio is stable across steps and closely matches the theory on both datasets.

SNR Ratio and Its Implications. Taking the ratio of the two SNRs (Eq. 17 and Eq. 20), the noise schedule terms $\bar{\alpha}_t$ and $(1-\bar{\alpha}_t)$ cancel exactly, yielding a ratio that depends only on the item similarity $c_{ji_\star}$ and is independent of the diffusion step t:

$$\frac{\mathrm{SNR}_{\mathrm{emb}}}{\mathrm{SNR}_{\mathrm{dist}}} = \sqrt{1-c_{ji_\star}} \leq 1, \quad \text{if } c_{ji_\star} \geq 0. \tag{21}$$

This demonstrates that embedding-space diffusion introduces an additional source of ranking noise, beyond the noise inherent to the diffusion process itself. This result has several important implications:

1. The degradation is universal. For any pair of items with positive inner product ($c_{ji_\star} > 0$), embedding-space diffusion has strictly lower ranking SNR than distribution-space diffusion. No choice of noise schedule, number of diffusion steps, or item set size can close this gap, as it is a structural property of the two approaches.

2. The degradation is worst where it matters most. In a learned embedding space, items that are most confusable with the target (those ranked just below it in the recommendation list) tend to have the highest inner product $c_{ji_\star}$. These items are precisely the ones whose ranking accuracy is critical to recommendation quality, and where the SNR degradation $\sqrt{1-c_{ji_\star}}$ is most severe. For example, for highly similar items with $c_{ji_\star} \geq 0.8$ (typical among the top-ranked distractors), the embedding-space diffusion preserves less than half of the ranking SNR of distribution-space diffusion exactly where it matters most for recommendation.

3. Connection to reverse process difficulty. Lower SNR in the forward process means that rankings are more severely corrupted at each step, introducing an additional burden beyond the noise inherent

to diffusion. The reverse process must recover the original concentrated preferences from these more heavily corrupted rankings. This is both a harder generation task (starting from worse inputs) and a harder learning task (the denoising network must learn to reconstruct fine-grained ranking distinctions from inputs where those distinctions have been systematically erased). With finite model capacity, this additional burden can lead to larger prediction errors that accumulate across denoising steps, ultimately degrading recommendation quality. In contrast, distribution-space diffusion preserves rankings longer, reducing this extra burden and facilitating accurate preference recovery.

Empirical Verification. We verify the theoretical SNR predictions on two representative datasets: Amazon Beauty and MovieLens-1M. The item embeddings used in this verification are from a trained PREDIFF model. For each dataset, we sample 500 test users and measure the pairwise ranking SNR between the target item and its top-10 most similar distractors (highest $c_{ji_\star}$) across all diffusion steps $t \in \{1, \ldots, 32\}$, averaging over 50 independent noise samples per step.

Figure 5 shows the results. On both datasets, the empirical SNR curves closely follow the theoretical predictions from Theorem 1 across all diffusion steps, with distribution-space diffusion consistently achieving higher SNR than embedding-space diffusion. The empirical ratios on Beauty (≈ 0.54) and ML-1M (≈ 0.51) closely match the theoretical values of 0.535 and 0.489, respectively. The bottom row further shows that the empirical ratio remains relatively stable across steps and closely matches the theoretical predictions, confirming that the SNR gap depends only on inter-item similarity $\bar{c}_{i\star}$ and is independent of the diffusion step t, as predicted by Theorem 1.

B Analysis of Soft Target Construction

In §4.2, we introduced the soft target used during training. Our soft target assigns probability λ to the target item and distributes $(1-\lambda)$ according to embedding similarity, encoding inter-item relationships into the training signal. This section provides additional analysis of this design.

Gradient Structure under Soft vs. One-Hot Targets. Under the KL divergence loss $\mathcal{L} = D_{\mathrm{KL}}(x_0 \| \hat{x}_0)$ with predicted distribution $\hat{x}_0 = \mathrm{softmax}(z)$, where z denotes the decoder's pre-softmax logits, the gradient with respect to z_j is:

$$\frac{\partial \mathcal{L}}{\partial z_j} = \hat{x}_{0,j} - x_{0,j}. \tag{22}$$

Under a *one-hot target* ($x_0 = \mathrm{OneHot}(i)$), this becomes:

$$\frac{\partial \mathcal{L}}{\partial z_j} = \begin{cases} \hat{x}_{0,i} - 1 & \text{if } j = i, \\ \hat{x}_{0,j} & \text{if } j \neq i. \end{cases} \tag{23}$$

All non-target items have target value $x_{0,j} = 0$, so the gradient is determined solely by the model's current prediction $\hat{x}_{0,j}$, with no distinction between similar and dissimilar items.

Under a *soft target* ($x_0 = \lambda \cdot \mathrm{OneHot}(i) + (1-\lambda) \cdot \mathrm{softmax}(\mathbf{s}_i/\tau)$), the gradient becomes:

$$\frac{\partial \mathcal{L}}{\partial z_j} = \hat{x}_{0,j} - \left[\lambda \cdot [\![j{=}i]\!] + (1-\lambda) \cdot \mathrm{softmax}(\mathbf{s}_i/\tau)_j\right]. \tag{24}$$

Items similar to the target now receive a smaller downward push, while dissimilar items receive a larger one. Concretely, let $p_j = \mathrm{softmax}(\mathbf{s}_i/\tau)_j$ denote the similarity weight for item j. Under the one-hot target, every non-target item receives gradient $\hat{x}_{0,j}$, pushing its score down equally. Under the soft target, a similar item j with large p_j receives a reduced gradient $\hat{x}_{0,j} - (1-\lambda)p_j$, while a dissimilar item with $p_j \approx 0$ still receives approximately $\hat{x}_{0,j}$. The model thus learns that predicting a similar item is less wrong than predicting an unrelated one, encoding fine-grained ranking structure into the gradient signal.

The similarity score $\mathbf{s}_i$ in Eq. 11 comes from the encoder's learned embeddings. Since the encoder and decoder share the embedding table and are trained jointly, better embeddings produce more informative soft targets, which in turn improve the decoder and yield better encoder gradients.

Similarity-Based Smoothing vs. Uniform Smoothing. Uniform label smoothing $\mathbf{y}_u = \lambda \cdot \mathrm{OneHot}(i) + (1-\lambda)/N$ treats every non-target item as equally irrelevant, transmitting no neighborhood information. Similarity-based smoothing, by contrast, encodes which specific items are close to the target, providing directional guidance that uniform smoothing cannot.

Role of λ and τ. The mixing weight λ interpolates between pure one-hot ($\lambda = 1$) and pure similarity-based targets ($\lambda = 0$). In practice, $\lambda = 0.8$ consistently works best across our tested four datasets (Figure 3(a)), suggesting that a strong target-item signal augmented with moderate neighborhood context provides the optimal balance.

The temperature τ controls the breadth of the neighborhood. As $\tau \to 0$, the similarity distribution sharpens to a peak at the most similar item, approaching a hard neighbor target. As $\tau \to \infty$, it flattens to uniform, recovering standard label smoothing. The experiments in Figure 3(b) show that $\tau \in [0.1, 0.2]$ works best, indicating that a soft but concentrated neighborhood distribution strikes the right balance.

C Dataset Statistics

We evaluate on four public benchmarks that span different domains and scales. Amazon Beauty and Toys and Games [McAuley et al., 2015] are e-commerce datasets with short interaction sequences and large item catalogs. MovieLens-1M and MovieLens-25M [Harper and Konstan, 2016] are movie rating datasets with longer histories per user. Table 3 summarizes their statistics. Following prior work [Kang and McAuley, 2018, Li et al., 2024], we use 5-core filtering for Amazon datasets and keep users with at least 5 interactions for MovieLens datasets.

Table 3: **Dataset statistics.** Sparsity = $1 - \frac{\#\text{Interactions}}{\#\text{Users} \times \#\text{Items}}$. Amazon Beauty and Toys and Games are the smaller-scale e-commerce datasets, while MovieLens are the larger-scale movie rating datasets with longer user histories.

	Beauty	**Toys and Games**	**MovieLens-1M**	**MovieLens-25M**
#Users	22,363	19,412	6,040	162,541
#Items	12,101	11,924	3,706	59,047
#Interactions	198,502	167,597	1,000,209	25,000,095
Avg. Length	8.88	8.63	165.60	153.81
Sparsity	99.93%	99.93%	95.53%	99.74%

D Implementation Details

All methods are trained with the Adam optimizer, a learning rate of 0.001, and a batch size of 512 for up to 300K steps, with early stopping (patience 10) on the validation metric. We provide the detailed configurations and training objectives of each method below.

- **PREDIFF.** The sequence encoder uses 2 Transformer layers with hidden dimension D=32 on Amazon datasets and D=128 on ML-1M, increasing to 4 layers with D=128 on ML-25M for large-scale datasets. The diffusion decoder uses 4 MLP layers for all datasets. We fixed diffusion steps T=32 with a linear noise schedule and λ=0.8 for the soft target. The distribution dropout rate is tuned over $\{0.3, 0.5, 0.7, 0.8, 0.9, 0.95\}$ and the temperature τ over $\{0.05, 0.1, 0.2, 0.5\}$.
- **SASRec [Kang and McAuley, 2018].** The hidden size is 128 on Amazon datasets and 256 on MovieLens. We use 2 Transformer blocks on all datasets except ML-25M, where we increase to 4 blocks. Training uses binary cross-entropy with sampled negatives.
- **BERT4Rec [Sun et al., 2019].** The hidden size is 64 on Amazon datasets and 256 on MovieLens, with 2 Transformer blocks. The mask ratio is set to 0.2 during training. The masked-item prediction objective uses full-catalog cross-entropy.
- **CL4SRec [Xie et al., 2022].** The hidden size is 64 on Amazon datasets and 256 on MovieLens. We use 2 Transformer blocks on most datasets and 4 on ML-25M. Training combines cross-entropy with sampled negatives and a contrastive loss with weight λ=0.1.

- **DiffRec [Wang et al., 2023].** We use the T-DiffRec variant, which represents the interaction sequence as a time-weighted multi-hot vector and trains the denoiser with timestep-reweighted mean squared error (MSE). The MLP hidden dimension is 1000, and the timestep embedding dimension is 10. The diffusion process uses $T{=}5$ steps under a linear-variance schedule.
- **DreamRec [Yang et al., 2023].** The hidden size is 64 on Amazon datasets and 256 on MovieLens, with 2 Transformer blocks on most datasets and 4 on ML-25M. The diffusion process uses $T{=}500$ steps with a linear noise schedule. The guidance weight is $w{=}2.0$ and the classifier-free dropout is $p{=}0.1$. The original model uses only an MSE objective. We found this MSE-only objective yielded poor recommendation performance in our experiments, therefore we added an auxiliary full-catalog cross-entropy objective on the sequence encoder.
- **CaDiRec [Cui et al., 2024].** The hidden size is 64 on Amazon datasets and 256 on MovieLens. We use 2 layers on most datasets and 4 on ML-25M. Following the official implementation, the diffusion process uses $T{=}1000$ steps with a linear noise schedule and 2 denoiser layers. The recommendation loss is binary cross-entropy with sampled negatives. The total objective also includes a contrastive loss with weight $\alpha{=}0.1$ and a denoising NLL loss with weight $\beta{=}0.2$, implemented as full-catalog cross-entropy at masked positions. The denoising MSE weight is $\gamma{=}0$, and the mask ratio is 0.1.
- **DiffuRec [Li et al., 2024].** The hidden size is 128 on Amazon datasets and 256 on MovieLens, with 4 Transformer blocks. The diffusion process uses $T{=}32$ steps with a linear noise schedule. Training uses full-catalog cross-entropy. The uncertainty parameter $\lambda{=}0.001$.
- **ADRec [Chen et al., 2025].** The hidden size is 128 on Amazon datasets and 256 on MovieLens, with 2 decoder blocks. The diffusion process uses $T{=}32$ steps with a linear noise schedule. Training follows a three-stage protocol of SASRec pretraining, warmup with frozen item embeddings, and joint training. The encoder shares the same configuration as SASRec. Pretraining uses binary cross-entropy with sampled negatives. Warmup and joint training use an equally weighted sum of full-catalog cross-entropy and denoising MSE over valid target positions.

Table 4 reports the number of trainable parameters for each method. PREDIFF has the fewest parameters among all diffusion-based methods, requiring less than half the parameters of the best-performing baseline.

Table 4: **Model size comparison**.

Method	Beauty	Toys and Games	MovieLens-1M	MovieLens-25M
SASRec	1.76M	1.73M	1.79M	16.75M
BERT4Rec	894K	883K	2.65M	16.87M
CL4SRec	878K	867K	2.06M	17.28M
DiffRec	24.23M	23.87M	7.43M	118.17M
DreamRec	890K	879K	2.78M	18.26M
CaDiRec	968K	956K	3.47M	18.43M
DiffuRec	2.47M	2.45M	4.63M	18.80M
ADRec	2.47M	2.45M	4.63M	20.38M
PREDIFF	486K	481K	2.03M	9.38M

E Extended Experiments

E.1 Additional Hyperparameter Analysis

We evaluate the impact of two additional hyperparameters, noise schedule and diffusion steps T, in Figure 6.

For noise schedule, we evaluate five schedules and find that the linear schedule performs best on both datasets, but the difference is relatively small, suggesting that the distribution-space formulation is not sensitive to how noise is injected over time.

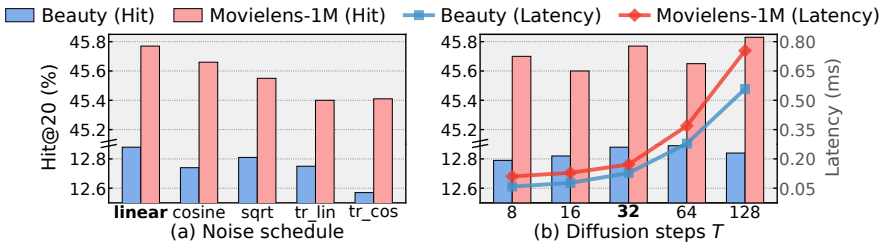


Figure 6: **Additional hyperparameter analysis.** (a) Noise schedule. (b) Diffusion steps T with inference latency (ms/sample). Default values in **bold**.

For diffusion steps T, performance generally improves as T increases, but the gains diminish beyond T=32 while inference latency continues to grow linearly. We therefore use T=32 as default, which achieves a good balance between accuracy and efficiency.

E.2 Additional Baseline Comparisons

Table 5 compares PREDIFF with the VAE-based method SVAE [Sachdeva et al., 2019] and the SID-based method TIGER [Rajput et al., 2023]. The results show that PREDIFF consistently outperforms both baselines, highlighting the effectiveness of the distribution-space diffusion approach.

Table 5: **Additional baseline comparisons (%).** All results are averaged over 3 random seeds with standard deviations reported. Best in **bold**. PREDIFF outperforms VAE-based and SID-based methods.

Method	Beauty				ML-1M			
	Hit@10	Hit@20	NDCG@10	NDCG@20	Hit@10	Hit@20	NDCG@10	NDCG@20
SVAE	$3.68_{\pm.07}$	$5.27_{\pm.10}$	$2.21_{\pm.05}$	$2.60_{\pm.06}$	$22.83_{\pm.12}$	$31.69_{\pm.13}$	$12.91_{\pm.18}$	$15.14_{\pm.15}$
TIGER	$4.57_{\pm.12}$	$6.82_{\pm.25}$	$2.44_{\pm.04}$	$3.00_{\pm.07}$	$19.61_{\pm.18}$	$30.34_{\pm.32}$	$10.38_{\pm.14}$	$12.94_{\pm.17}$
PREDIFF	$\mathbf{9.44}_{\pm.13}$	$\mathbf{12.88}_{\pm.26}$	$\mathbf{5.68}_{\pm.03}$	$\mathbf{6.55}_{\pm.05}$	$\mathbf{34.74}_{\pm.17}$	$\mathbf{45.77}_{\pm.23}$	$\mathbf{20.43}_{\pm.14}$	$\mathbf{23.20}_{\pm.08}$

E.3 Additional Ablation Study

Table 6 separates the effects of the training objective, soft targets, and distribution-space diffusion. Starting from the original SASRec [Kang and McAuley, 2018], which uses binary cross-entropy (CE) with sampled negatives, we replace its objective with full-category softmax CE and then add soft targets. PREDIFF variant without diffusion retains its sequence encoder and soft targets but predicts item scores directly by the encoder, achieving results comparable to SASRec with full-softmax CE and soft targets.

Table 6: **Additional ablation study (%).** Distribution-space diffusion produces the largest performance gain while the training recipe such as full-category softmax CE and soft targets further improves performance.

Settings	Beauty		ML-1M	
	Hit@20	NDCG@20	Hit@20	NDCG@20
SASRec (original)	$7.94_{\pm.05}$	$3.47_{\pm.02}$	$34.38_{\pm.30}$	$15.50_{\pm.09}$
+ full-softmax CE	$8.32_{\pm.03}$	$4.26_{\pm.04}$	$37.61_{\pm.09}$	$17.49_{\pm.13}$
+ full-softmax CE + soft targets	$8.63_{\pm.05}$	$4.47_{\pm.03}$	$38.07_{\pm.06}$	$17.78_{\pm.01}$
PREDIFF w/o diffusion	$8.37_{\pm.13}$	$4.24_{\pm.05}$	$37.68_{\pm.39}$	$17.60_{\pm.13}$
PREDIFF (full)	$\mathbf{12.88}_{\pm.26}$	$\mathbf{6.55}_{\pm.05}$	$\mathbf{45.77}_{\pm.23}$	$\mathbf{23.20}_{\pm.08}$

These results indicate that distribution-space diffusion provides the largest performance improvement, while the training techniques such as full-category softmax CE and soft targets offer additional gains by providing richer supervision and better capturing neighborhood information.

E.4 Case Study

We present qualitative case studies from four datasets to illustrate how PREDIFF's distribution-space formulation captures user preferences that embedding-space baselines miss. For each dataset, we present two test users and show the top-5 recommendations from PREDIFF, DiffuRec [Li et al., 2024], and ADRec [Chen et al., 2025].

In each case, history items that provide key signals toward the target are marked in **blue bold**, and the target item's rank under each method is shown next to the method name.

Amazon Beauty — Brand Loyalty (Fekkai Hair Care)

Recent History: Seki Edge Fingernail Clipper → Covergirl Mousse Mascara → NEXXUS Platinum Shampoo → Lysol Dish Soap Dispenser → Dove Body Wash → Zwilling Multi-Use Tool → Sarna Sensitive Lotion → **Fekkai Essential Shea Shampoo** → **Fekkai Brilliant Glossing Shampoo**

PreDiff rank = 1	DiffuRec rank = 91	ADRec rank = 7052
1. ★ **Fekkai Glossing Conditioner**	1. John Frieda Foam Color	1. Olay Regenerist UV Cream
2. Fekkai Prx Shampoo	2. Dove Leave-In Foam	2. Clean & Clear Blackhead Kit
3. Fekkai Hair Spray	3. Fekkai Color Conditioner	3. Dead Sea Mud Soap
4. Fekkai Color Care Conditioner	4. Suave Leave-In Foam	4. Henna Bright Red Powder
5. Fekkai Volume Shampoo	5. Hyaluronic Moisturizing Cream	5. Nicole by OPI Nail Polish

Note: The two consecutive Fekkai shampoo purchases signal strong brand loyalty. PREDIFF captures this and recommends exclusively Fekkai products. DiffuRec recommends hair care from mixed brands with only one Fekkai item, while ADRec diverges entirely into skincare and nail polish.

Amazon Beauty — Fragrance Preference

Recent History: Body Drench Tanning Accelerator → **Angel by Thierry Mugler EDP** → **D&G Light Blue EDT** → Savon de Marseille Soap

PreDiff rank = 2	DiffuRec rank = 9497	ADRec rank = 6230
1. Pink Sugar by Aquolina EDT	1. Dabur Vatika Coconut Oil	1. Afirm 3X Treatment
2. ★ **Fancy by Jessica Simpson**	2. Olay Microdermabrasion Kit	2. Reb'L Fleur by Rihanna EDP
3. M by Mariah Carey EDP	3. Clairol Hair Color	3. Revlon Emery Boards
4. White Shoulders EDP	4. L'Oreal Sleek Serum	4. Conair Flat Iron
5. Redken Shampoo Set	5. Moroccan Oil Treatment	5. Salicylic Acid 30% Peel

Note: The history contains two designer fragrances (Angel by Thierry Mugler and D&G Light Blue), indicating a fragrance preference. PREDIFF recommends four perfumes in its top-5. DiffuRec recommends hair oils and skincare with no fragrance items, while ADRec includes one perfume but otherwise recommends unrelated hair tools and chemical peels.

Amazon Toys and Games — Nerf Blaster Collector

Recent History: **Double Shot Rifle** → **Nerf N-Strike Recon CS-6** → **Nerf N-Strike Raider CS-35** → **Nerf N-Strike Longstrike CS-6** → **Nerf Rapid Fire 20**

PreDiff rank = 3	DiffuRec rank = 214	ADRec rank = 9658
1. Nerf N-Strike Stampede	1. Nerf N-Strike Firefly	1. Say Anything Family
2. Nerf Dart Ammo Box	2. Halo ODST Soldier	2. WowWee Robosapien
3. ★ **Nerf Dart Tag Speedload 6**	3. Nerf N-Strike Stampede	3. LEGO City Camper
4. Nerf Raider CS-35	4. Nerf N-Strike Deploy	4. Nerf Nite Finder 2-Pack
5. Nerf Longstrike CS-6	5. Nerf N-Force Sword	5. ALEX Pirates of the Tub

Note: All five history items are Nerf blasters, revealing a dedicated collector. PREDIFF recommends five Nerf products. DiffuRec partially captures the brand but includes an unrelated Halo action figure and a melee foam sword rather than blasters, while ADRec recommends board games, robots, and LEGO sets.

Amazon Toys and Games — Super Mario Fan

Recent History: Fisher-Price Imaginext Batcave → Fisher-Price Learning Basketball → Imaginext DC Super Friends 5-Pack → **Super Mario Yoshi Red Plush**

PreDiff rank = 1	DiffuRec rank = 59	ADRec rank = 5871
1. ★ **Super Mario Luigi Plush**	1. LEGO Blue Building Plate	1. LEGO Blue Building Plate
2. LEGO Star Wars X-Wing	2. Angry Birds Board Game	2. Orb Factory Mosaics
3. Angry Birds Star Wars	3. LEGO City Police Forest	3. SmartLab Squishy Head
4. Imaginext Gotham City Jail	4. Super Mario Boo Plush	4. Fisher-Price Dollhouse
5. Angry Birds Star Wars Leia	5. Barbie Hair Salon	5. LeapFrog Pet Pals 2

Note: The recent Yoshi plush purchase signals interest in the Super Mario franchise. PREDIFF immediately recommends the Luigi plush from the same series. DiffuRec includes one Mario plush but buries it among LEGO sets and a Barbie salon, while ADRec recommends entirely unrelated items (mosaics, dollhouses, educational games).

MovieLens-1M — Romantic Drama

Recent History: What Dreams May Come → St. Elmo's Fire → Practical Magic → 'Til There Was You → Stanley & Iris → Flashdance → Up Close and Personal → Simply Irresistible → Carrington

PreDiff rank = 3	DiffuRec rank = 1571	ADRec rank = 23
1. How Stella Got Her Groove Back	1. Bring It On	1. Home for the Holidays
2. Random Hearts	2. Golden Bowl	2. Nell
3. ★ **Dying Young (1991)**	3. Pillow Book	3. Jane Eyre
4. Mad Love	4. Farewell My Concubine	4. Princess Caraboo
5. Beautician and the Beast	5. Dreaming of Joseph Lees	5. Now and Then

Note: All nine history items are romantic dramas, indicating a strong genre preference. PREDIFF recommends five romantic dramas. DiffuRec recommends art-house films from different genres (Pillow Book, Farewell My Concubine). ADRec's top-5 includes dramas but lacks the romantic tone of the history.

MovieLens-1M — Classic Film & War

Recent History: Silence of the Lambs → Shawshank Redemption → Schindler's List → 2001: A Space Odyssey → Fargo → GoodFellas → Wizard of Oz → Like Water for Chocolate

PreDiff rank = 1	DiffuRec rank = 130	ADRec rank = 284
1. ★ **Apocalypse Now (1979)**	1. Good Will Hunting	1. Leaving Las Vegas
2. Bridge on the River Kwai	2. Apollo 13	2. Titanic
3. Babe	3. Babe	3. Pulp Fiction
4. Amadeus	4. Sling Blade	4. Sense and Sensibility
5. Henry V	5. October Sky	5. Babe

Note: The history is dominated by critically acclaimed classics (Schindler's List, Fargo, GoodFellas, 2001). PREDIFF recommends films of similar caliber including Apocalypse Now, Bridge on the River Kwai, and Amadeus. Both baselines lean toward lighter, more mainstream titles (Good Will Hunting, Apollo 13, Titanic, Sense and Sensibility).

MovieLens-25M — Documentary Enthusiast

Recent History: 42 Up → Harlan County U.S.A. → Brother's Keeper → One Day in September → When We Were Kings → Crumb → Dear Zachary → Man on Wire → Devil and Daniel Johnston → Silence of the Lambs

PreDiff rank = 1	DiffuRec rank = 21	ADRec rank = 1994
1. ★ **Paradise Lost (1996)**	1. Shawshank Redemption	1. The Fugitive
2. Murderball	2. Sixth Sense	2. Shawshank Redemption
3. Jesus Camp	3. American Beauty	3. Pulp Fiction
4. Enron: Smartest Guys	4. Forrest Gump	4. Forrest Gump
5. Hoop Dreams	5. Jurassic Park	5. American Beauty

Note: Nine of ten history items are documentaries, spanning true-crime (Brother's Keeper), sports (When We Were Kings), and biographical films (Man on Wire). PREDIFF recommends five documentaries. Both baselines ignore the documentary preference entirely, recommending popular fiction films (Shawshank Redemption, Forrest Gump, Jurassic Park).

MovieLens-25M — War Film Fan

Recent History: Stalag 17 → Dirty Dozen → M*A*S*H → Aliens → Star Wars: Return of the Jedi → Catch-22 → Good Morning, Vietnam → Bat*21 → Kelly's Heroes → Incredible Mr. Limpet

PreDiff rank = 3	DiffuRec rank = 1642	ADRec rank = 876
1. Red Dawn	1. Dirty Rotten Scoundrels	1. Johnny Dangerously
2. Independence Day	2. Bill & Ted's Adventure	2. Ice Age
3. ★ **Heartbreak Ridge (1986)**	3. City Slickers	3. Down and Out in Beverly Hills
4. Starship Troopers	4. Bad News Bears	4. Usual Suspects
5. Tora! Tora! Tora!	5. Strange Brew	5. Scotland, Pa.

Note: Seven of ten history items are war films (Stalag 17, Dirty Dozen, M*A*S*H, Catch-22, Good Morning Vietnam, Bat*21, Kelly's Heroes). PREDIFF recommends war and military-themed films (Red Dawn, Tora! Tora! Tora!). DiffuRec recommends comedies (City Slickers, Bad News Bears), and ADRec recommends a mix of comedies and thrillers, both missing the war genre signal.

F Broader Impacts

This work improves the accuracy of sequential recommendation systems, which can help users discover relevant content with less effort. On the other hand, more accurate recommendation may reinforce filter bubbles by narrowing the diversity of suggested items. We encourage practitioners to combine our method with diversity-aware re-ranking strategies to mitigate this risk. Our method does not involve personal data collection beyond standard interaction logs already used by existing recommenders, and we do not foresee direct negative societal consequences beyond those common to recommendation systems in general.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our three contributions (distribution-space diffusion with ranking preservation guarantee, sparse projection, and soft targets) and the scope of experimental validation on four benchmarks. The claims are supported by theoretical analysis (§3) and experiments (§5).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations at the end of the paper after the conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Theorem 1 is stated with assumptions in §3, and a complete proof is provided in Appendix A with empirical verification.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All hyperparameters, training details, model architectures, and dataset preprocessing are described in §5 and Appendix D. Code is available at `https://github.com/Komorebi660/PreDiff`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Source code and instructions to reproduce experiments are available at `https://github.com/Komorebi660/PreDiff`. All datasets used are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training details including optimizer, learning rate, batch size, data splits (leave-one-out), and hyperparameter ranges are specified in §5 and Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The results in Table 1 and Table 2 report mean and standard deviation over 3 random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments are conducted on NVIDIA V100 GPUs (§5). Training PREDIFF takes approximately 1 hour on Beauty/Toys, 2 hours on ML-1M, and 24 hours on ML-25M per run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research uses only publicly available benchmark datasets and does not involve human subjects, deception, or any violation of the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in Appendix F, including potential benefits (improved user experience) and risks (filter bubbles) with suggested mitigation strategies.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our method is a recommendation algorithm that does not pose high risks for misuse comparable to language models or image generators.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and baseline methods are properly cited. Amazon Reviews (`https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html`) is released for research use with citation required; no formal license is specified. MovieLens (`https://grouplens.org/datasets/movielens/`) is released for non-commercial research use under its own terms of use. Our usage complies with both datasets' terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release our code at `https://github.com/Komorebi660/PreDiff` with a README documenting usage, dependencies, and reproduction steps.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a component of our core method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.