

Unlocking the Power of Diffusion Models in Sequential Recommendation: A Simple and Effective Approach

Jialei Chen

MIC Lab,

College of Computer Science and
Technology,
Jilin University
Changchun, China
chenjl21@mails.jlu.edu.cn

Yuanbo Xu*

MIC Lab,

College of Computer Science and
Technology,
Jilin University
Changchun, China
yuanbox@jlu.edu.cn

Yiheng Jiang

MIC Lab,

College of Computer Science and
Technology,
Jilin University
Changchun, China
jiangyh22@mails.jlu.edu.cn

Abstract

In this paper, we focus on the often-overlooked issue of embedding collapse in existing diffusion-based sequential recommendation models and propose ADRec, an innovative framework designed to mitigate this problem. Diverging from previous diffusion-based methods, ADRec applies an independent noise process to each token and performs diffusion across the entire target sequence during training. ADRec captures token interdependency through auto-regression while modeling per-token distributions through token-level diffusion. This dual approach enables the model to effectively capture both sequence dynamics and item representations, overcoming the limitations of existing methods. To further mitigate embedding collapse, we propose a three-stage training strategy: (1) pre-training the embedding weights, (2) aligning these weights with the ADRec backbone, and (3) fine-tuning the model. During inference, ADRec applies the denoising process only to the last token, ensuring that the meaningful patterns in historical interactions are preserved. Our comprehensive empirical evaluation across six datasets underscores the effectiveness of ADRec in enhancing both the accuracy and efficiency of diffusion-based sequential recommendation systems.

CCS Concepts

- Information systems → Recommender systems.

Keywords

Diffusion Model; Sequential recommendation

ACM Reference Format:

Jialei Chen, Yuanbo Xu, and Yiheng Jiang. 2025. 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 V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada*. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3711896.3737172>

*corresponding author

[†]The ADRec code is available at <https://github.com/Nemo-1024/ADRec>.

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

KDD '25, Toronto, ON, Canada

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1454-2/2025/08

<https://doi.org/10.1145/3711896.3737172>

KDD Availability Link:

The source code of this paper has been made publicly available at <https://doi.org/10.5281/zenodo.15470542>.

1 Introduction

Sequential recommendation (SR) has long been a cornerstone of modern recommendation systems. Its primary objective is to analyze historical interaction records between users and the system to predict the next item a user is likely to engage with. [19, 26]. Various sequence models, such as Recurrent Neural Networks (RNNs) and Transformers [25], have been widely adopted in SR. Transformer-based methods, which leverage the powerful self-attention mechanism [25], including SASRec [12] and BERT4Rec [22], have emerged as some of the most effective models for SR. Despite these advancements, sequential recommendation still faces significant challenges, such as weak representation spaces [17, 28, 30].

Diffusion models [10, 21] may be regarded as a distinctive form of self-supervised learning due to their inherent diffusion process [2, 6, 17, 29]. They have exhibited notable capabilities in representation learning, which could prove invaluable in accurately capturing the distribution of item representations within recommendation systems. Nevertheless, their efficacy within recommendation systems has been comparatively disappointing in recent years, failing to yield the anticipated advancements. This elicits a critical inquiry: *Are diffusion models fundamentally ill-suited for recommendation systems, or do they encounter intrinsic limitations in this domain?*

We hypothesize that the issue arises from **embedding collapse in diffusion-based methods**. As shown in Figure 1, the Transformer-based SASRec+ [13] forms a structured yet narrow and fragile representation space, which lacks robustness—a limitation shared by other traditional recommendation models. Diffusion-based methods like DiffuRec [16] and DreamRec [30] yield representation spaces that are not only narrow but also resemble isotropic Gaussian distributions, indicating a severe collapse. This resemblance is particularly concerning, as randomly initialized embeddings also follow such distributions, suggesting that these models fail to learn meaningful representations beyond their initial state.

To tackle the embedding collapse, we first summarize the architectural design and training strategies of existing methods in Table 1 and point out several potential flaws that could impact the use of diffusion models in sequential recommendation:

- **Only performing denoising learning on the last item is insufficient.** Existing methods, including DiffuRec, DimeRec [15], and DreamRec, focus on diffusion and compute loss only

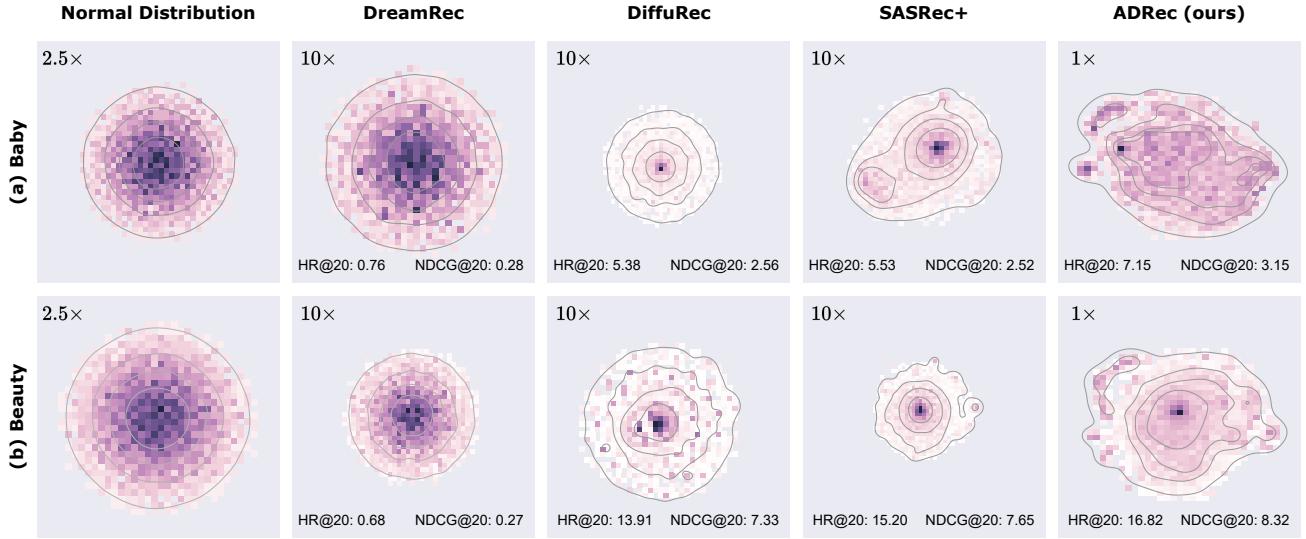


Figure 1: T-SNE results of the learned item embeddings of ADRec and other baselines on the Baby and Beauty dataset. If the contour shape closely resembles isotropic Gaussian noise (as seen in DreamRec and DiffuRec) or if the representation space is narrow (as observed in DreamRec, DiffuRec, and SASRec+, which requires a large magnification factor), it suggests a weak embedding space. In contrast, ADRec maintains a structured embedding space and expands it compared to SASRec+, significantly enhancing item separability. Additional visualization results can be found in Appendix Figure 10.*

for the final target item instead of for the entire one-position offset target sequence. As a result, only a limited number of items are subjected to diffusion, significantly complicating the process of learning item distributions and contributing to embedding collapse. From an auto-regressive perspective, they miss the chance to implement per-token teacher forcing, which restricts the effectiveness of sequence modeling.

- **Inappropriate training loss impacts the effectiveness of diffusion models.** Some works, like DreamRec, use only denoising loss (Mean Squared Error, MSE) as the training objective. This method does not align with the recommendation task and often produces suboptimal results. DiffuRec [16] retains recommendation loss (Cross-Entropy, CE) as the training objective but does not incorporate denoising loss, which limits the ability to exploit the potential of diffusion models fully.
- **End-to-end training leads to Embedding collapse.** In the original application scenarios of diffusion models, input features (from images, audio, etc.) are fixed, meaningful, and distinguishable from one another. Item embeddings initialized from scratch are random and lack significance [7], which creates a risk of embedding collapse, where all embedding weights converge to the same value to make the denoising learning easier. This collapse can cause the denoising loss to become zero during training, leading to poor performance during the inference phase.
- **The sequence-level diffusion process causes a mismatch during the inference phase.** It affects the model’s inference

by introducing noise throughout the entire sequence, which can corrupt the historical sequence during inference. This, in turn, impacts the quality of the target item prediction during inference.

To tackle these challenges, we introduce **ADRec (Auto-regressive Diffusion Recommendation model)**, a novel approach that merges a token-level diffusion process with causal attention for denoising. Unlike earlier diffusion-based methods, ADRec applies independent noise levels to each token within the entire one-position offset target sequence. This method guarantees that every token in the sequence undergoes diffusion, enabling the model to better learn item distributions while maintaining the per-token teacher-forcing feature of auto-regression. By capturing item dependencies through auto-regression and concurrently learning the distribution of each item via diffusion, ADRec can more effectively model both sequence dynamics and item representations.

To further mitigate embedding collapse, we introduce a three-stage training strategy. In the first stage, we pre-train the embedding weights using a causal attention module that helps ensure the embedding space is structured before full parameter training begins. During this stage, the model is similar to SASRec+ and employs CE loss. In the second stage, we warm up the backbone of ADRec to align its parameters with the pre-trained embedding weights, freezing the embedding weights during this phase. Finally, in the third stage, we conduct full-parameter training of ADRec. The last two stages employ a combination of CE and MSE to align the recommendation task while fully utilizing the distribution modeling capabilities of the diffusion process to optimize the embedding weights, ultimately creating a robust and structured embedding space (see last column in Figure 1).

*Embeddings are normalized before T-SNE to ensure that the scale of the visualization results is comparable. “10×” indicates that the visualization results have been magnified tenfold. In this context, “Normal Distribution” serves as a baseline, representing randomly initialized embedding weights.

During the inference phase, the token-level diffusion process applies denoising iterations solely to the last target while assigning the diffusion time steps for the other positions to zero. The independent noise process prevents the introduction of unreasonable noise into historical interactions, enabling ADRec to concentrate on meaningful patterns within the sequence.

To this end, ADRec overcomes all the previously mentioned limitations. We empirically demonstrate its effectiveness across six datasets of varying scales. The experimental results show that ADRec significantly outperforms mainstream baselines, achieving improvements of 15.45% and 13.02% on HR@20 and NDCG@20, respectively. Despite employing a three-stage training framework, ADRec reduces the training time by an average of 70.98% compared with the best diffusion baseline.

2 Background

We have previously discussed several SR methods, and because of space limitations, we have included the Related Works in Appendix Section A. Before presenting our model, we briefly introduce the standard diffusion model and sequential recommendation as foundational knowledge.

2.1 Diffusion Models

Diffusion models [10, 21] have gained popularity as generative modeling techniques in various domains. They can be seen as a specific method of self-supervised learning that utilizes a diffusion process to enhance representation learning.

Forward Process In this paper, we denote the diffusion process with the subscript t to indicate diffusion time steps. For each token x , we define a forward diffusion process that progressively adds Gaussian noise to the data over a series of time steps. This process is modeled as a Markov chain, where the data at each step k is incrementally noised.

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

where \mathcal{N} is the normal distribution, and β_t is the variance of the noise added at each step controlled by a schedule $\{\beta_k \in (0, 1)\}_{k=1}^K$. The process continues until the data is converted into pure noise at x_T . With $\bar{\alpha}_t = \prod_{t' \leq t} (1 - \beta_{t'})$, we can analytically write the result of the forward process given an original data:

$$\begin{aligned} q(x_t | x_0) &= \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I) \\ i.e. \quad x_t &= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{(1 - \bar{\alpha}_t)}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \end{aligned} \quad (2)$$

$\bar{\alpha}_t$ is the noise schedule employed for the forward process. In original DDPM [10], t is randomly sampled from a uniform distribution $\mathcal{U}[0, 1]$ and a function maps t to $\bar{\alpha}_t \in [0, 1]$.

Reverse Process The reverse process is also a Markov chain and attempts to recreate the original data from the noise with a denoising model $p_\theta(x_{t-1} | x_t)$:

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

where the mean μ_θ is modeled with a neural network, and one can set the covariance to the identity scaled by a fixed constant depending on t . The μ_θ can be formulated by either using the noise

ϵ or the target x_0 :

$$\mu_\theta = \frac{1}{\sqrt{1 - \beta_t}}x_t - \frac{\beta_t}{\sqrt{(1 - \beta_t)(1 - \bar{\alpha}_t)}}\epsilon \quad (4)$$

$$= \frac{\sqrt{1 - \beta_t}(1 - \bar{\alpha}_t))}{1 - \bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_0 \quad (5)$$

In Equation (4), the model learns to predict the noise $\hat{\epsilon}$; while in Equation (5), the model learns to predict the original data \hat{x}_0 . We use the latter formulation. Thus, a denoising model $f_\theta(x_t, t)$ can be trained to predict the original data x_t that is available at timestamp t . Efficient training of DMs is possible by optimizing the simplified MSE loss instead of the original variational lower bound (VLB) as

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{t, x_0, \epsilon} \left[\|x_0 - f_\theta(x_t, t)\|_2^2 \right] \quad (6)$$

2.2 Sequential Recommendation

Given an interaction sequence $S = [i^0, i^1, \dots, i^{L-1}, i^L]$, where i^k denotes the k -th interacted item and L indicates the maximum sequence length in the training set, the aim of SR is to produce a ranked list of items as predicted candidates for the next item the user is likely to interact. There are generally two auto-regressive training strategies for sequential recommendation models.

The first strategy, referred to as *one-step prediction*, leaves one item out, using $i_{tgt} = i^L$ as the target sequence. The model learns the mapping $g_\theta(i^{0:L-1}) = i^L$, where the model predicts the next item in the sequence. The second strategy, called *per-step prediction*, shifts the sequence by one position, using $i_{tgt} = i^{1:L}$ as the target sequence. This strategy employs per-token teacher forcing, where the model predicts each item in the sequence based on the previous item: $g_\theta(i^k) = i^{k+1}$ for $0 \leq k < L$. Both strategies share the same inference process.

3 Methodology

3.1 Auto-regressive strategy

In the previous section, we examined several auto-regressive training strategies that are essential for model architecture, as they directly affect the tensor shape of the target sequence. Let the target embedding sequence be denoted as \mathbf{x} . For one-step prediction, $\mathbf{x} = e^L \in \mathbb{R}^{B \times 1 \times D}$, and for per-step prediction, $\mathbf{x} = e^{1:L} \in \mathbb{R}^{B \times L \times D}$, where B is the batch size, L is the sequence length, and D is the embedding dimension. For simplicity, we ignore sequence padding in this discussion. We adopt a per-token prediction strategy, ensuring that each output token contributes to the loss calculation and thus benefits from the improved sequence mining capabilities provided by per-token teacher forcing.

Existing diffusion-based SR algorithms typically employ one-step prediction, where denoising learning is only applied to the final target item, e^L [3, 15, 16, 28, 30]. This approach has two main limitations. First, one-step prediction undermines the model's capacity to capture correlations between tokens, as it loses the per-token teacher forcing. Second, the available samples for diffusion learning become excessively sparse. For example, in the MovieLens-100K (ML-100K) dataset, there are a total of 1,008 items but only 938 sequences, which means that only 938 samples are utilized for denoising learning in one-step prediction. From an individual standpoint,

Table 1: A comparison of the architectural design and training strategies between ADRec and prior works.

	Diffusion Modelling	Loss Computation	Train Objective	Denoising Model	Noise Process	Embed.Pre-train	Embed.Collapse
DiffRec[27] (SIGIR '23)	full sequence	per token	MSE	MLP	sequence-level	✗	✓
DreamRec[30] (NeurIPS '23)	only last token	only last token	MSE	MLP	sequence-level	✗	✓
DiffuRec[16] (TOIS '23)	only last token	only last token	CE	Causal Attention	sequence level	✗	✓
DimeRec[15] (WSDM '24)	only last token	only last token	CE + MSE	MLP	sequence-level	✓	✓
ADRec (ours)	per token	CE + MSE	Causal Attention	token-level	✓	✗	

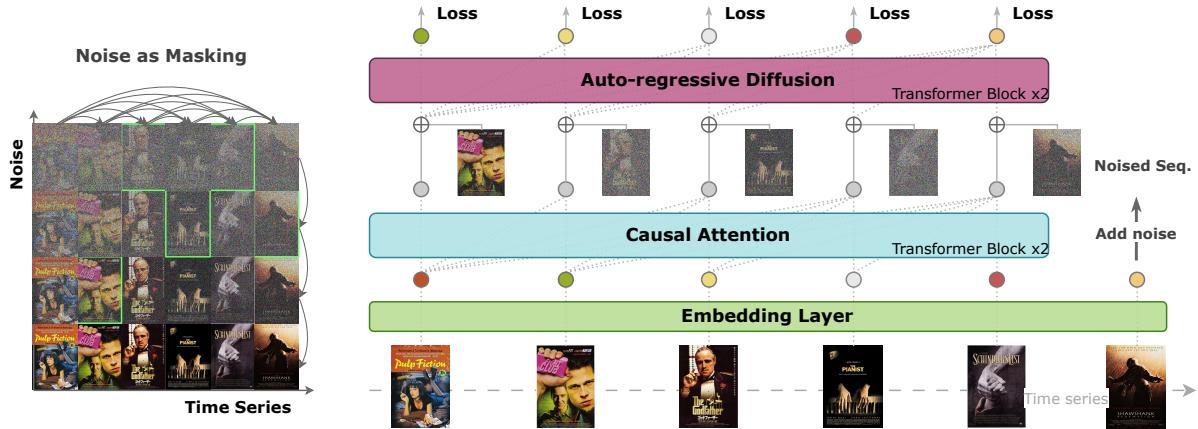


Figure 2: Method overview. The left diagram illustrates that time series and diffusion processes are two orthogonal directions of evolution, with noise acting as a soft mask to measure uncertainty. ADRec applies independent diffusion processes to individual items, with the noise level of the items in the current target sequence highlighted in green box.[†]

this sparsity inhibits the effective learning of item distributions. From a broader perspective, it fails to sufficiently represent the entire embedding space, resulting in embedding collapse.

DiffuRec attempts to tackle this issue by partitioning the original target sequence into subsequences based on temporal order, giving each item in the sequence the chance to model the distribution. However, this strategy considerably increases both the number of sequences and the training time required. Essentially, it is similar to performing per-step prediction but without utilizing the parallelization capabilities of the self-attention mechanism, resulting in significantly longer training times.

3.2 ADRec Architecture

This section provides a detailed description of ADRec’s architectural design. ADRec employs a multi-layer Transformer encoder at its backbone, which has demonstrated strong performance and versatility in modeling sequential dependencies. The architecture consists of two key components: the causal attention module and the auto-regressive diffusion module, each built using two Transformer encoder layers to maximize generality and to simplify the design (right in Figure 2). *ADRec models the interdependence of tokens by auto-regression, jointly with the per-token distribution by token-level diffusion.*

3.2.1 Causal Attention Module. As illustrated in Figure 2, we define an embedding sequence $\mathbf{e} \in \mathbb{R}^{B \times L \times D}$, where B represents the batch size and D denotes the hidden size, as the semantic encoding

of the intrinsic latent aspects captured by the interacted sequence $\mathbf{i} \in \mathbb{R}^L$. The primary role of causal attention module (CAM), parameterized as $CAM\phi(\cdot)$, is to extract historical sequence information $\mathbf{c} \in \mathbb{R}^{B \times L \times D}$ that has not yet been corrupted by noise and use it as conditional guidance for the Auto-regressive Diffusion Module (ADM) during denoising learning.

$$\mathbf{c} = CAM\phi(\mathbf{e}) \quad (7)$$

Notably, we do not incorporate positional encoding. Experimental results (Appendix Table 8) indicate that including positional encoding leads to a slight decrease in performance, as further discussed in Appendix Section D.

3.2.2 Feature Aggregation. Before introducing the auto-regressive diffusion module, it is essential to define an appropriate feature aggregation method for the conditional guidance $\mathbf{c} \in \mathbb{R}^{B \times L \times D}$, the noised target sequence $\mathbf{x}_t \in \mathbb{R}^{B \times L \times D}$, and the current diffusion time step $t \in \mathbb{R}^{B \times L}$.

We first scale the diffusion time step and encode it using an MLP with a SiLU activation function:

$$\mathbf{t}_{emb} = MLP\left(\frac{1000 \cdot t}{T}\right) \quad (8)$$

Next, we aggregate the various components like DiffuRec [16] and denote the result as \mathbf{z} :

$$\mathbf{z} = \mathbf{c} + \lambda \odot (\mathbf{x}_t + \mathbf{t}_{emb}) \quad (9)$$

where λ is a coefficient, and we set $\lambda = 1e^{-3}$, which was the optimal setting in DiffuRec. We also experimented with replacing linear aggregation with cross-attention, but the results were unsatisfactory (see Figure 5 in Experiments).

[†]The movie posters in Figure 2 are for illustrative purposes only; during actual training, ADRec only utilizes user interaction data, like movie ID.

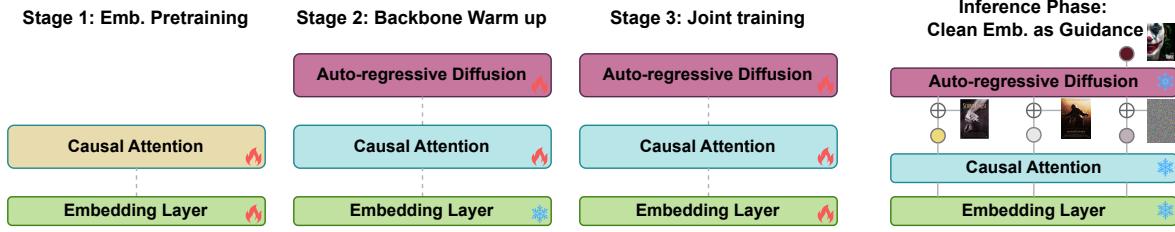


Figure 3: The three-stage training strategy and the inference strategy of ADRec.

3.2.3 *Auto-regressive Diffusion Module.* At this point, we introduce the auto-regressive diffusion module(ADM), parametrized as $\text{ADM}_\theta(\cdot)$. The ADM module consists of a two-layer Transformer encoder, which also serves as the denoising model for the diffusion process.

$$\hat{\mathbf{x}} = \text{ADM}_\theta(\mathbf{z}) \quad (10)$$

where $\hat{\mathbf{x}}$ is the reconstructed target sequence.

Token-independent Diffusion Process We propose a token-independent diffusion process, which independently applies the noise process to each token. We can present the diffusion time steps as: $\mathbf{t} \in \mathbb{R}^{B \times L}$. This approach fundamentally differs from classic sequence-level diffusion models, which use a uniform noise process across all tokens in a sequence ($\mathbf{t} \in \mathbb{R}^B$). We denote the token-level diffusion process as:

$$\mathbf{x}_{t_i}^i = \sqrt{\bar{\alpha}_{t_i}^i} \mathbf{x}_0^i + \sqrt{(1 - \bar{\alpha}_{t_i}^i)} \epsilon, \quad \epsilon \sim N(0, I) \quad (11)$$

where $1 \leq i \leq L$ and $0 \leq t \leq T$, with T representing the maximum diffusion step. For different indices i_1 and i_2 , the noise process can vary ($t_{i_1} \neq t_{i_2}$). We denote the noisy sequence as \mathbf{x}_t for convenience.

Chen et al. [1] demonstrated that the token-independent diffusion process optimizes a re-weighting of an Evidence Lower Bound (ELBO) on the expected log-likelihoods $\ln p_\theta((\mathbf{x}_{t_i}^i)_{1 \leq i \leq L})$, where the expectation is averaged over noise levels $t_{1:L} \sim [T]^L$. The advantage of the token-level diffusion process stems from the cross-perspective between the diffusion process and masked generative models [14] views noise as a soft mask, interpreting the entire diffusion process as a sequence of soft masking operations (as shown on the left in Figure 2). Consequently, even if a few prior tokens are corrupted significantly, the model can still effectively learn to sample from the correct conditional distribution, capturing the distribution of all possible subsequences in the training set. This approach offers potential benefits in terms of model robustness and debiased learning. Furthermore, the token-independent diffusion process provides unique advantages during the inference phase, which we explore in Section 3.5.

3.3 Training Objective

In Section 3.1, we adopted the per-step prediction auto-regressive strategy, which necessitates computing the training loss for each output target token. To this end, we use both the recommendation loss (Cross-Entropy, CE, $\mathcal{L}_{ce}(\cdot)$) and the denoising loss (Mean Squared Error, MSE, $\mathcal{L}_{mse}(\cdot)$) as joint training objectives, which is beneficial for both the recommendation task and embedding distribution modeling. For the recommendation loss, we keep it simple by using full-item CE loss without negative sampling.

We first compute the similarity score $\mathbf{s} \in \mathbb{R}^{B \times L \times N}$, where N is the total number of items in the dataset, by taking the inner product of the reconstructed embedding $\hat{\mathbf{x}}$ and the transpose of the entire item embedding $\mathbf{E} \in \mathbb{R}^{N \times D}$, as follows:

$$\mathbf{s} = \text{Softmax}(\hat{\mathbf{x}} \odot \mathbf{E}^\top) \quad (12)$$

Next, we calculate the total loss \mathcal{L} as:

$$\mathcal{L} = \mathcal{L}_{ce}(\mathbf{s}, \mathbf{i}_{tgt}) + \mathcal{L}_{mse}(\hat{\mathbf{x}}, \mathbf{x}) \quad (13)$$

Some works, such as DreamRec, have explored using only the denoising loss as the training objective [18, 30]. However, these approaches often fail to achieve satisfactory results because denoising loss does not directly align with the recommendation task. On the other hand, approaches like DiffuRec have addressed this by retaining recommendation loss as the primary training objective while omitting the denoising loss. However, such a strategy significantly constrains the potential of diffusion models in sequential recommendation.

3.4 Training Strategy

Section 1 mentioned the severe embedding representation collapse issue. Here, we present a three-stage training strategy to mitigate this problem effectively (see Figure 3).

In the first stage, we pre-train the embedding layer using the causal attention module, with the training objective being cross-entropy loss. Our goal is to obtain a semantically rich and structured embedding space. In the second stage, we conduct a 5-epoch warm-up on the backbone of ADRec, freezing the embedding weights. The causal attention module is trained from scratch. The goal is to align the denoising model with the embedding space. Without this alignment, early-stage gradient updates could degrade the pre-trained embedding. Finally, we conduct full-parameter training, leveraging the powerful representation learning capabilities of the diffusion model to optimize the embedding weights further. Experimental results demonstrate that our training method effectively expands the representational capacity of the embedding space and increases the distinction between items. We did not observe any signs of representation collapse in ADRec.

3.5 Inference Strategy

Existing methods typically employ classical sequence-level diffusion processes, introducing unnecessary noise to the historical sequence during the inference phase. The goal of the denoising model is to reconstruct the clean embedding. Therefore, during the inference phase, noise should only be applied to the last position, while the embeddings at other positions remain clean. However,

Table 2: Main results (%) on six datasets. The best results are in boldface, and the second-best are underlined. Improv. is the relative improvement of the best method against the second-best one.

Dataset	Metric	GRU4Rec	BERT4Rec	LightSANS	FEARec	SASRec+	EulerFormer	CORE	SVAE	DreamRec	DiffuRec	ADRec	Improv.
Baby	HR@20	5.4576	4.0009	4.0658	3.5786	5.5268	5.7906	2.7472	2.7439	0.7648	5.382	7.1524	23.52%
	NDCG@20	2.2568	1.6198	1.499	1.513	2.5197	<u>2.4982</u>	0.9427	1.0086	0.2777	<u>2.5649</u>	3.1455	22.64%
Beauty	HR@20	12.6462	9.8723	9.5874	9.7294	<u>15.2038</u>	14.7346	7.5635	3.9204	0.6815	13.9102	16.8246	10.66%
	NDCG@20	5.6347	3.9216	4.5894	4.3869	<u>7.6509</u>	7.5428	2.6558	1.4985	0.2728	7.3326	8.3214	8.76%
ML-100K	HR@20	18.6858	10.1833	14.1129	9.6959	18.4538	<u>19.3524</u>	11.8659	8.1504	3.4395	16.0688	22.0699	14.04%
	NDCG@20	7.1357	3.7504	5.0443	3.5048	7.1034	<u>7.6313</u>	4.0904	3.2905	1.3339	6.555	9.0028	17.97%
Sports	HR@20	6.0511	4.255	4.4246	3.9261	6.4059	<u>6.4759</u>	2.8127	2.0019	0.7432	6.2174	8.1639	26.07%
	NDCG@20	2.6069	1.7844	1.8463	1.6689	<u>3.344</u>	3.2259	0.9802	0.8447	0.2148	3.2067	3.6389	8.82%
Toys	HR@20	5.6216	4.9423	4.3054	6.2075	10.7082	<u>10.912</u>	6.1481	1.5031	0.4755	9.859	12.0924	10.82%
	NDCG@20	2.5605	1.9627	1.8942	3.3492	6.0517	<u>6.2564</u>	2.217	0.6266	0.1731	5.8508	6.7982	8.66%
Yelp	HR@20	6.3762	4.0681	3.7219	3.005	6.6894	6.435	2.3459	1.9238	0.7681	<u>6.7315</u>	7.2433	7.60%
	NDCG@20	2.5408	1.5224	1.4837	1.0783	2.5277	2.4252	0.8788	0.7804	0.2477	<u>2.5951</u>	2.8875	11.27%

for the one-step prediction strategy (shown in Appendix Figure 8), during inference, the noised target $\mathbf{x}_t \in \mathbb{R}^{B \times 1 \times D}$ is broadcast across the sequence length dimension to align with the conditional guidance $\mathbf{c} \in \mathbb{R}^{B \times L \times D}$, inadvertently introducing noise into the historical sequence and degrading the model’s inference performance. The situation remains problematic for the per-step prediction strategy. The sequence-level diffusion process requires the noise process to remain consistent (shown in the middle of Appendix Figure 8). The uniform noise process $\mathbf{t} = [t, t, \dots, t, t]$ is applied to all target tokens ($\mathbf{x}^{1:L}$) from the maximum diffusion step T down to zero.

A common compromise is to decouple the sequence modeling from the diffusion process. Specifically, a sequence model is first employed to capture the correlations among historical items, and the resulting representations are then passed into a diffusion module, where a non-sequence model, such as an MLP, acts as the denoising model [15, 30]. However, relying solely on an MLP for denoising eliminates the opportunity to leverage shared attention mechanisms between the historical sequence and the denoising target. Such mechanisms could provide better guidance for denoising by dynamically directing attention based on historical interactions.

The token-level diffusion process in ADRec provides substantial advantages during the inference phase. Specifically, it allows us to apply pure noise only to the last token position while setting the noise schedule for the other positions to zero (as shown on the right in Figure 3 and Appendix Figure 8). This approach ensures that, during inference, ADRec can perform diffusion iterations on the target item while still receiving guidance from the clean historical sequence at each time step via attention. The time steps during the inference phase are represented as follows:

$$\begin{aligned} \mathbf{t} &= [0, 0, \dots, 0, T] \\ \mathbf{x}_t &= [x_0^1, x_0^2, \dots, x_0^{L-1}, x_t^L] \end{aligned} \quad (14)$$

4 Experiments

4.1 Experimental Settings

4.1.1 Datasets. We evaluate the effectiveness of ADRec using six commonly used and publicly available datasets: **Baby**, **Beauty**, **Sports**, **Toys**, **ML-100K**, and **Yelp**. We follow the common pre-processing steps outlined in [12, 16] to ensure a minimum of 5

Table 3: Training complexity comparison. L is the sequence length and d is the hidden size.

Model	GRU4Rec	SASRec+	DreamRec	DiffuRec	ADRec
Complexity	$O(Ld^2)$	$O(Ld^2 + dL^2)$	$O(Ld^2 + dL^2)$	$O(Ld^2 + dL^2)$	$O(Ld^2 + dL^2)$

interactions associated with each user and item. Statistics of the preprocessed datasets are summarized in Appendix Table 7.

4.1.2 Baselines. To comprehensively demonstrate the effectiveness of ADRec, we select target models from various categories, as follows:

- **RNN-based:** GRU4Rec [ICLR ’16] [9].
- **Transformer-based:** BERT4Rec [22] (CIKM ’19); LightSANS [5] (SIGIR ’21); FEARec [4] (SIGIR ’23); SASRec+ [13] (RecSys ’23); EulerFormer [23] (SIGIR ’24).
- **Embedding-enhancement:** CORE [11] (SIGIR 22 short).
- **VAE-based:** SVAE [20] (WSDM ’19).
- **Diffusion-based:** DreamRec [30] (NeurIPS ’23); DiffuRec [16] (TOIS ’23).

4.1.3 Evaluation Protocol. Similar to the setup in most existing works, we partition the users into training, validation, and test sets in a ratio of 8:1:1. We evaluate all methods using two widely adopted metrics: Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG). HR reflects the model’s retrieval ability, while NDCG indicates its ranking performance. In all experiments, we report the results as percentages.

4.1.4 Implementation Details. We set the maximum number of training epochs to 500, with a validation interval of 5 epochs. Early stopping is applied if the evaluation metric does not improve on the validation set for four consecutive epochs. The training batch size is fixed at 512; the Adam optimizer has a learning rate of 1×10^{-3} and a cosine annealing schedule. The embedding size and hidden size are set to 128, and the maximum sequence length is set to 50 for all datasets. We use cross-entropy loss for all items without negative sampling as the recommendation loss for all baselines. For diffusion-based baselines, we use the *truncated linear* schedule for

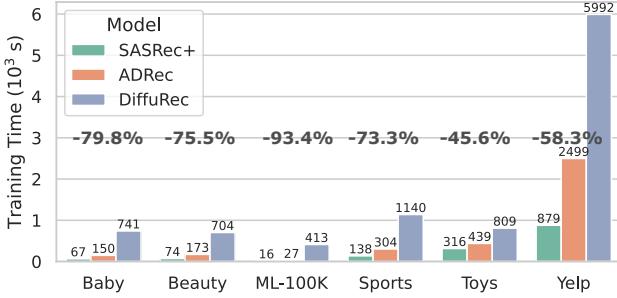


Figure 4: Comparison of training time between ADRec and other methods. "-xx%" indicates the reduction in training time for ADRec compared to DiffuRec.

the noise process, with 32 diffusion steps, following [16]. Experimental results show that ADRec is not sensitive to the number of diffusion steps (see Appendix Figure 9). We evaluated each method over five independent runs and reported the average results.

4.2 Overall Performance

In this section, we evaluate the performance of ADRec by comparing it against ten baseline models across six datasets. The results are presented in Table 2.

ADRec consistently outperforms all baselines, demonstrating its superior performance. Specifically, compared to the best-performing baselines, ADRec achieves average improvements of 15.45% on HR@20 and 13.02% on NDCG@20. These significant improvements underscore the effectiveness of ADRec in capturing three critical aspects: maintaining a structured embedding space without collapse, modeling token interdependency through auto-regression, and jointly modeling per-token distributions through diffusion.

Notably, compared to SASRec+, which closely resembles the pre-training stage of ADRec (stage 1 in Figure 3), the superiority of ADRec lies in its distribution modeling capability enabled by the diffusion model. In contrast, DreamRec and DiffuRec suffer from embedding collapse due to inadequate architecture design and training strategies, with performance often falling below that of SASRec+.

4.3 Complexity and Training Time Analysis.

The time complexity of ADRec is solely determined by its Transformer backbone, i.e., $O(nd^2 + n^2d)$, with the diffusion process introducing no additional overhead during training, thus matching the complexity of most baselines, as shown in Table 3. During inference, the time complexity of all diffusion-based methods becomes $O(T \times (nd^2 + n^2d))$, where T denotes the number of diffusion steps.

For training time, we report the actual GPU wall-clock time from the start of training to early stopping. Stage 1 of ADRec has a duration similar to SASRec+. Stage 2 lasts for only 5 epochs, resulting in a very short training time. Stage 3 requires approximately the same amount of time as Stage 1. As a result, the overall training time of ADRec is approximately twice that of SASRec+.

DiffuRec computes the loss only on the last token of each sequence, sacrificing the parallelism of the Transformer. To cover

Table 4: Results (%) of ablation experiments conducted for ADRec on six datasets.

Dataset	Metric	ADRec	w/o warm-up	w/o pre-train	w/o mse	w/o CAM	MLP
baby	HR@20	7.1524	6.7507	4.5217	4.0439	7.0320	6.8216
	NDCG@20	3.1455	2.9535	2.1488	1.9483	3.0853	2.8186
beauty	HR@20	16.8246	15.7483	13.233	12.6733	16.0798	16.0738
	NDCG@20	8.3214	8.1056	6.9819	6.7209	7.9479	8.0568
ml-100k	HR@20	22.0699	21.0144	14.835	13.2913	18.9609	20.0558
	NDCG@20	9.0028	8.3206	5.5249	5.3358	7.5569	7.8465
sports	HR@20	8.1639	7.2156	5.9635	5.6443	8.0550	7.6604
	NDCG@20	3.6389	3.7413	3.0097	2.9709	3.5146	3.1831
toys	HR@20	12.0924	11.9226	9.3835	8.2456	12.0264	11.4461
	NDCG@20	6.7982	6.6113	5.4178	4.8172	6.5298	6.1759
yelp	HR@20	7.2433	6.9927	7.0224	5.6989	4.9116	6.9563
	NDCG@20	2.8875	2.8597	2.7823	2.2123	1.9548	2.4569

all interactions, DiffuRec splits a sequence of length n into $n-1$ subsequences, where each interaction serves as the last token of a subsequence. This process expands the dataset size by a factor of n , leading to an approximate n -fold increase in training time per epoch.

Despite using a three-stage training strategy, ADRec reduces total training time by an average of 70.98% across the six datasets compared to DiffuRec (see Figure 4).

4.4 Ablation Study

To evaluate the effectiveness of each design choice in ADRec, we perform ablation studies with four variants:

- (Variant 1) **w/o warm-up**: ADRec without warmup stage (training stage 2).
- (Variant 2) **w/o pre-train**: ADRec trained in an end-to-end manner without pre-training.
- (Variant 3) **w/o MSE**: End-to-end ADRec without the denoising loss $\mathcal{L}_{mse}(\hat{x}, x)$.
- (Variant 4) **w/o CAM**: ADRec removed CAM module.
- (Variant 5) **MLP**: ADRec with an MLP as the denoising model.

Table 4 presents the ablation results across six datasets. Comparing the original ADRec with Variant 1 reveals that aligning pre-trained embeddings with the model backbone is crucial for performance. The significant performance drop, compared to Variant 2, emphasizes the importance of the three-stage training strategy, which provides a meaningful and structured embedding space to the diffusion module and mitigates embedding collapse. In contrast to Variant 3, we find that jointly optimizing the denoising loss and recommendation loss is essential for effectively deploying the diffusion model, leading to improved recommendation performance. Without CAM, compared with Variant 4, ADRec can still extract historical interaction information through self-attention, but a performance drop is observed, especially on ML-100K and Yelp. Finally, when comparing with Variant 5, which employs an MLP for denoising, we observe that even with the same MLP architecture used in DreamRec, ADRec's careful architectural design and training strategy still ensure superior performance. In comparison, DreamRec's performance is nearly equivalent to random recommendation, underscoring the effectiveness of our approach.

Table 5: Comprehensive evaluation of embedding representations in the original embedding space.

Dataset	Model	Top 5 Singular Values	Singular Value Variance ↑	Singular Value Entropy ↓	Covariance Matrix Entropy ↓	Isotropy Score ↓	KL Div. to Gaussian ↑
Baby	DreamRec	[80.58, 80.09, 79.91, 79.02, 78.75]	35.29	4.85	4.83	0.51	0.96
	DiffuRec	[106.31, 90.62, 85.69, 81.53, 81.36]	66.94	4.85	4.82	0.13	1.83
	SASRec+	[84.74, 79.90, 79.44, 78.92, 78.59]	38.89	4.85	4.84	0.24	1.09
	ADRec	[247.88, 164.00, 145.34, 138.33, 130.06]	1017.59	4.74	4.33	0.02	10.38
Beauty	DreamRec	[89.70, 88.48, 88.32, 87.74, 87.62]	33.16	4.85	4.84	0.56	0.70
	DiffuRec	[129.98, 116.93, 113.33, 97.61, 97.35]	108.99	4.84	4.81	0.10	2.25
	SASRec+	[128.70, 125.14, 106.41, 97.86, 94.20]	101.55	4.84	4.81	0.09	2.11
	ADRec	[177.19, 159.37, 145.52, 137.74, 126.75]	995.55	4.76	4.51	0.02	10.37
ML-100K	DreamRec	[43.16, 42.64, 42.42, 41.83, 41.47]	33.56	4.84	4.78	0.23	4.41
	DiffuRec	[55.42, 49.44, 44.24, 42.55, 41.41]	38.75	4.83	4.77	0.04	5.15
	SASRec+	[47.37, 42.03, 41.78, 41.60, 41.50]	39.33	4.83	4.78	0.11	4.60
	ADRec	[90.91, 82.50, 71.20, 63.54, 53.68]	122.57	4.80	4.55	0.04	10.43
Sports	DreamRec	[122.18, 121.81, 121.40, 121.04, 120.88]	33.16	4.85	4.85	0.66	0.35
	DiffuRec	[220.33, 182.85, 172.72, 151.13, 137.70]	347.81	4.84	4.79	0.09	3.43
	SASRec+	[133.91, 125.74, 122.40, 122.00, 120.98]	68.15	4.85	4.84	0.13	0.81
	ADRec	[415.16, 355.94, 317.40, 275.30, 257.72]	433.00	4.65	3.97	0.01	10.37
Toys	DreamRec	[109.57, 107.46, 106.19, 104.88, 103.48]	91.02	4.85	4.82	0.40	1.59
	DiffuRec	[139.38, 135.46, 117.65, 106.74, 105.77]	138.12	4.84	4.81	0.10	2.39
	SASRec+	[103.28, 97.70, 96.78, 96.15, 95.16]	52.63	4.85	4.84	0.13	1.03
	ADRec	[182.03, 156.40, 148.03, 139.77, 137.68]	1243.76	4.75	4.51	0.01	10.37
Yelp	DreamRec	[265.59, 265.08, 264.78, 264.31, 264.19]	32.51	4.85	4.85	0.84	0.06
	DiffuRec	[772.81, 690.32, 623.17, 587.23, 577.26]	13798.54	4.75	4.27	0.03	10.36
	SASRec+	[843.78, 675.20, 636.04, 625.65, 568.57]	12385.48	4.77	4.30	0.03	10.36
	ADRec	[810.29, 781.18, 625.42, 566.81, 541.51]	18955.47	4.68	4.12	0.01	10.36

Table 6: Linear Probe accuracy of different models on ML-100K. Each item is multi-class, with a total of 26 classes.

	DreamRec	DiffuRec	SASRec+	ADRec
Precision	20%	35%	42%	44%
Recall	14%	30%	40%	46%
F1-score	16%	31%	40%	46%

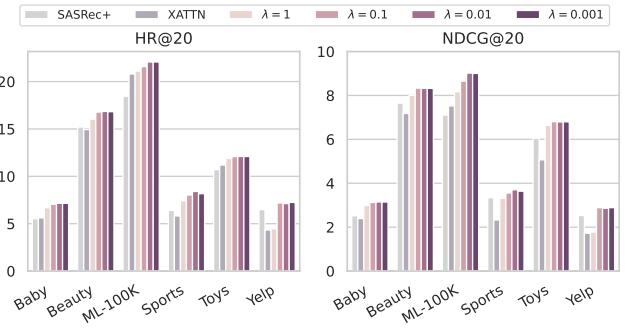
4.5 Embedding Collapse

4.5.1 t-SNE Visualization. Recent studies [6] have highlighted that diffusion models excel not only in continuous generation tasks but also in representation learning. To illustrate that ADRec can learn a more structured representation space, we visualize the embedding weights of ADRec, DiffuRec, DreamRec, and SASRec+ (the latter is also considered as ADRec in its pre-training stage) using T-SNE [24], as shown in Figure 1 and Appendix Figure 10.

We begin by focusing on the shapes of the visualizations. The embeddings of DiffuRec and DreamRec closely resemble isotropic standard Gaussian noise, particularly in their contour shapes, which indicate embedding collapse. While ADRec retains distinct structured features. More importantly, the scale of the representation space expanded clearly in ADRec. As indicated by the scale label in the top-left corner of the figure, ADRec effectively enlarges the weak representation space in the pre-train stage (similar to SASRec+), making the embeddings more distinguishable.

4.5.2 Comprehensive Evaluation of Embedding Collapse in Original Embedding Space. Beyond the t-SNE visualizations, we provide additional quantitative analyses in the original embedding space, as presented in Table 5:

- Top-5 singular values and singular value variance are much higher in ADRec, indicating strong principal component dominance and the ability to capture more discriminative features. Other models (especially DreamRec) show flatter singular values, implying weak feature directions.
- Singular value entropy, covariance entropy, and isotropy score are lower for ADRec, demonstrating more directional (anisotropic)

**Figure 5: Comparison of ADRec with linear integration using different λ coefficients and cross-attention integration, with SASRec+ as a baseline.**

embeddings and better parameter efficiency. This highlights another aspect of ADRec’s superior performance: ADRec does not just have more information; it has information concentrated in effective directions.

- Silhouette scores are also higher for ADRec, suggesting clearer clustering among item embeddings.

All these metrics indicate that the characteristics of the original embedding space for all DM based models align closely with the visualizations produced by t-SNE. DreamRec still resembles a Gaussian distribution, similar to randomly initialized embeddings, while ADRec exhibits the most anisotropy.

4.5.3 Linear Probing. We also conduct a linear probing experiment to further confirm the importance of the expanded and structured embedding space in ADRec. The experimental details are provided in Appendix Section C. From the results in Table 6, it is clear that ADRec achieves the best performance across all three metrics, whereas DiffuRec and DreamRec suffer from poor performance due to embedding collapse. Compared to SASRec+, the expanded embedding space of ADRec enhances the model’s ability to recognize embeddings, thereby improving its overall performance.

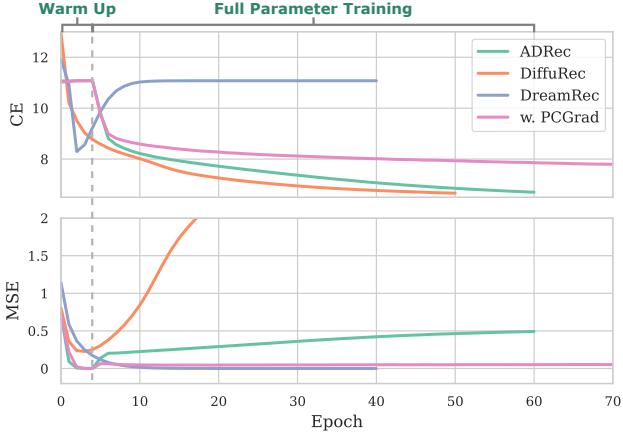


Figure 6: Training loss curves on the Yelp dataset. “w. PCGrad” denotes ADRec trained using the PCGrad framework. The pre-training stage of ADRec is omitted for clarity.

4.6 Feature Aggregation Method

In this section, we explore various feature aggregation methods. Specifically, as illustrated in Figure 5, we investigate the impact of varying λ coefficients in the linear integration method, following the approach used in DiffuRec. Our results indicate that ADRec demonstrates considerable robustness to changes in λ , with most settings outperforming SASRec+. A moderate enhancement in the efficacy of recommendations is noted as λ diminishes.

Notably, the training strategy of ADRec enhances the robustness of the embedding space, enabling the sequence model to tolerate greater noise introduced into the historical sequence without causing significant degradation of the embeddings. In contrast, DiffuRec is highly sensitive to λ , as highlighted in the original paper. For example, when $\lambda = 0.1$, performance declines by up to 80% compared to $\lambda = 0.001$, which worsens the collapse of DiffuRec’s weak embedding space. Furthermore, we experimented with feature integration using cross-attention; however, this approach resulted in decreased performance.

4.7 Joint Optimization Objective of ADRec

Diffusion-based recommendation models frequently encounter the challenge of selecting the appropriate training objective.

Figure 6 illustrates the training loss curve on Yelp. We observe that DiffuRec and DreamRec, which focus on a single objective, experience degradation in the other objective, adversely affecting overall recommendation performance. In contrast, for ADRec, we notice that losses display a distinct step-wise pattern throughout training. During the backbone warm-up stage, the convergence for the recommendation loss (CE) and denoising loss indicates alignment between the backbone network and the pre-trained embedding weights. In the full-parameter training phase, the recommendation loss continues to decline, suggesting further optimization of the embedding weights. Although the denoising loss shows a slight upward trend, it remains minimal compared to DiffuRec.

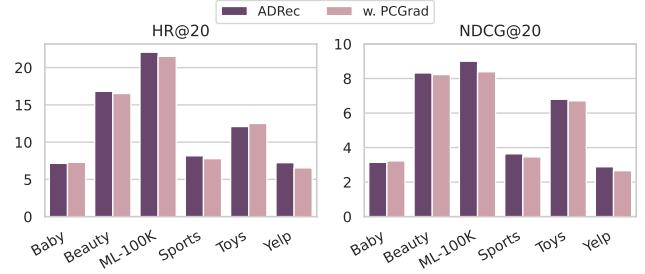


Figure 7: Performance impact of introducing the PCGrad framework on six datasets.

Li et al. [15] argues that a joint training objective of CE and MSE may lead to potential gradient conflicts. To verify this, we incorporate the PCGrad technique [31] into ADRec, which projects conflicting gradients to maintain an acute angle between their directions during optimization.

The joint optimization objective can be viewed as a trade-off between the generation and classification tasks. After incorporating the PCGrad framework, the degradation in denoising loss is notably suppressed; however, the recommendation performance does not reach the original ADRec level. And a slight degradation in recommendation performance is observed in Figure 7. In contrast to prior approaches, ADRec demonstrates only mild gradient conflict. While its optimization leans toward the recommendation objective, it maintains a meaningful level of denoising learning—unlike DiffuRec, which effectively neglects it. This balanced trade-off plays a key role in enhancing recommendation quality.

5 Conclusion

In this paper, we identify and address the embedding collapse problem in diffusion-based sequential recommendation models. We propose ADRec, a novel framework that integrates an auto-regressive strategy with a token-level diffusion process. By decoupling embedding optimization into semantic pretraining and subsequent denoising, the proposed three-stage training strategy effectively prevents embedding collapse that can occur when denoising is applied directly to randomly initialized embeddings. By tackling key challenges such as weak representation spaces and embedding collapse, ADRec significantly improves the model’s capacity to learn meaningful item distributions and sequence dynamics.

While our results demonstrate significant advancements, there remains considerable potential for further improvement in utilizing diffusion models for recommendation systems. ADRec provides a solid foundation for diffusion-based sequential recommendation methods, with opportunities for future enhancements.

Acknowledgments

This work is supported by the Natural Science Foundation of China No. 62472196, Jilin Science and Technology Research Project 20230101067JC, National Key R&D Program of China under Grant No. 2021ZD0112501 and 2021ZD0112502, National Natural Science Foundation of China under Grant No. 62272193, National Key R&D Program of China under Grant Nos. 2022YFB3103700 and 2022YFB3103702.

References

- [1] Boyuan Chen, Diego Martí Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. 2024. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion. doi:10.48550/arXiv.2407.01392 arXiv:2407.01392
- [2] Xinlei Chen, Zhuang Liu, Saining Xie, and Kaiming He. 2024. Deconstructing denoising diffusion models for self-supervised learning. *arXiv preprint arXiv:2401.14404* (2024).
- [3] Hanwen Du, Huanhuan Yuan, Zhen Huang, Pengpeng Zhao, and Xiaofang Zhou. 2023. Sequential Recommendation with Diffusion Models. arXiv:2304.04541 [cs.IR] <https://arxiv.org/abs/2304.04541>
- [4] Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Jianfeng Qu, Fuzhen Zhuang, Guanfeng Liu, and Victor S. Sheng. 2023. Frequency Enhanced Hybrid Attention Network for Sequential Recommendation. arXiv:2304.09184 [cs.IR] <https://arxiv.org/abs/2304.09184>
- [5] Xinyan Fan, Zheng Liu, Jianxun Lian, Wayne Xin Zhao, Xing Xie, and Ji-Rong Wen. 2021. Lighter and better: low-rank decomposed self-attention networks for next-item recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 1733–1737.
- [6] Michael Fuest, Pingchuan Ma, Ming Gui, Johannes S. Fischer, Vincent Tao Hu, and Björn Ommer. 2024. Diffusion Models and Representation Learning: A Survey. arXiv:2407.00783 [cs.CV] <https://arxiv.org/abs/2407.00783>
- [7] Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. 2024. Empowering Diffusion Models on the Embedding Space for Text Generation. arXiv:2212.09412 [cs.CL] <https://arxiv.org/abs/2212.09412>
- [8] Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634* (2022).
- [9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939 [cs.LG] <https://arxiv.org/abs/1511.06939>
- [10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*.
- [11] Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao. 2022. CORE: Simple and Effective Session-based Recommendation within Consistent Representation Space. In *SIGIR*.
- [12] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
- [13] Anton Klenitskiy and Alexey Vasilev. 2023. Turning Dross Into Gold Loss: is BERT4Rec really better than SASRec? In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1120–1125.
- [14] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. 2024. Autoregressive Image Generation without Vector Quantization. *NeurIPS 2024* (2024).
- [15] 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. 2024. DimeRec: A Unified Framework for Enhanced Sequential Recommendation via Generative Diffusion Models. doi:10.48550/arXiv.2408.12153 arXiv:2408.12153
- [16] Zihao Li, Aixin Sun, and Chenliang Li. 2023. DiffuRec: A Diffusion Model for Sequential Recommendation. doi:10.48550/arXiv.2304.00686 arXiv:2304.00686
- [17] Jianghao Lin, Jiaqi Liu, Jiachen Zhu, Yunjin Xi, Chengkai Liu, Yangtian Zhang, Yong Yu, and Weinan Zhang. 2024. A Survey on Diffusion Models for Recommender Systems. doi:10.48550/arXiv.2409.05033 arXiv:2409.05033 [cs]
- [18] Yong Niu, Xing Xing, Zhihun Jia, Ruidi Liu, Mindong Xin, and Jianfu Cui. 2024. Diffusion Recommendation with Implicit Sequence Influence. In *Companion Proceedings of the ACM Web Conference 2024 (Singapore, Singapore) (WWW '24)*. Association for Computing Machinery, New York, NY, USA, 1719–1725. doi:10.1145/3589335.3651951
- [19] Liwei Pan, Weike Pan, Meiyang Wei, Hongzhi Yin, and Zhong Ming. 2024. A Survey on Sequential Recommendation. arXiv:2412.12770 [cs.IR] <https://arxiv.org/abs/2412.12770>
- [20] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. 2019. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 600–608.
- [21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456* (2020).
- [22] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.
- [23] Zhen Tian, Wayne Xin Zhao, Changwang Zhang, Xin Zhao, Zhongrui Ma, and Ji-Rong Wen. 2024. EulerFormer: Sequential User Behavior Modeling with Complex Vector Attention. arXiv:2403.17729 [cs.IR] <https://arxiv.org/abs/2403.17729>
- [24] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
- [25] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).
- [26] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects. In *28th International Joint Conference on Artificial Intelligence, IJCAI 2019*. International Joint Conferences on Artificial Intelligence, 6332–6338.
- [27] Wenjie Wang, Yiyuan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. 2023. Diffusion Recommender Model. doi:10.48550/arXiv.2304.04971 arXiv:2304.04971
- [28] Yu Wang, Zhiwei Liu, Liangwei Yang, and Philip S. Yu. 2023. Conditional Denoising Diffusion for Sequential Recommendation. arXiv:2304.11433 [cs.LG] <https://arxiv.org/abs/2304.11433>
- [29] Weilai Xiang, Hongyu Yang, Di Huang, and Yunhong Wang. 2023. Denoising Diffusion Autoencoders are Unified Self-supervised Learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
- [30] Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. 2023. Generate What You Prefer: Reshaping Sequential Recommendation via Guided Diffusion. doi:10.48550/arXiv.2310.20453 arXiv:2310.20453
- [31] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems* 33 (2020), 5824–5836.

A Related Works

A.1 Traditional Sequential Recommendation

Recent sequential recommendation (SR) advancements have leveraged powerful deep neural networks to capture item interdependencies. GRU4Rec [9] employs a gated recurrent unit (GRU) to model the temporal dependencies within user sequences. SASRec [12] adopts a multi-layer transformer architecture to efficiently model sequence interactions. SASRec+ [13], an enhancement of SASRec, replaces the original binary cross-entropy loss with cross-entropy loss, significantly improving performance. BERT4Rec [22] extends this approach by incorporating bidirectional transformer layers and using a Cloze task to capture user sequence patterns better. LightSANS [5] introduces a low-rank decomposition to the self-attention mechanism, enhancing both the efficiency and effectiveness of transformer-based SR models. EulerFormer [23] introduces a unified theoretical framework to capture both semantic and positional differences between items in a transformer, enhancing the model's expressive power in sequence modeling. These methods have demonstrated strong sequence modeling capabilities and provide valuable insights for our work.

A.2 Diffusion Models for SR

In recent years, diffusion models have been increasingly applied to sequential recommendation, aiming to improve the quality of the embedding space. DiffuRec [27] falls within the collaborative recommendation domain, where it predicts whether a user will interact with each item. DreamRec [30] employs a diffusion model to explore the underlying distribution of target items, generating an oracle next-item embedding that aligns with user preferences, eliminating the need for negative sampling. DiffuRIS [18] enhances behavior sequence-based guidance representations by explicitly modeling both long- and short-term user interests. However, DiffuRec and DreamRec rely solely on denoising loss, which does not align with the recommendation task. DiffuRec [16], the most substantial open-source baseline, performs denoising only on the final target item, using cross-entropy loss. This approach limits the model's capacity to capture sequence dynamics and item distribution. DimeRec [15] introduces joint training by applying both denoising loss and

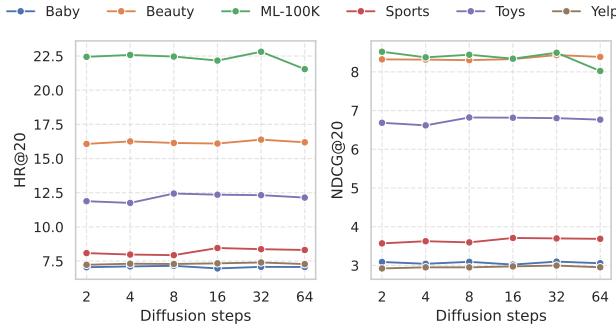


Figure 9: Recall@20 and NDCG@20 at different diffusion steps

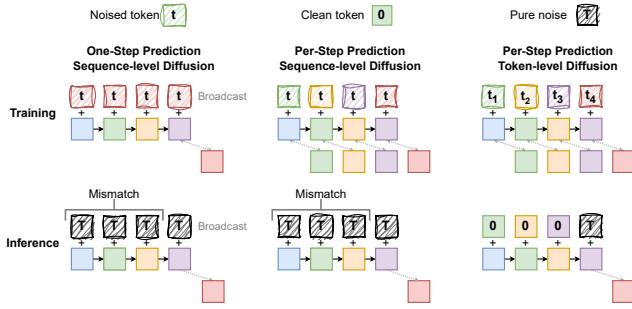


Figure 8: Diagrams of different auto-regressive strategies and diffusion strategies.

Table 7: Statistics of datasets after preprocessing.

	Baby	Beauty	ML-100K	Sports	Toys	Yelp
Users	11761	10553	938	22686	11268	136346
Items	4731	6086	1008	12301	7309	64669
Interactions	92613	94119	54457	185779	95468	1857033
Avg. length	7.89	8.92	58.01	8.19	8.47	13.62
Sparsity	99.62%	99.74%	94.50%	99.63%	99.95%	99.98%

Table 8: The impact of positional encoding on ADRec performance. "with PE" means ADRec with positional encoding.

Model	Metric	Baby	Beauty	ML-100K	Sports	Toys	Yelp
ADRec	HR@20	7.1524	16.8246	22.0699	8.1639	12.0924	7.2433
ADRec	NDCG@20	3.1455	8.3214	9.0028	3.6389	6.7982	2.8875
with PE	HR@20	7.0034	16.3726	21.0172	7.9949	11.5668	7.5212
with PE	NDCG@20	2.9843	7.8961	8.4448	3.2769	5.8444	3.1355

recommendation loss to the final output item and mitigates gradient conflicts in the joint loss by introducing noise to the final target item embedding using a complex geodesic random walk. However, existing diffusion-based methods rarely address the issue of embedding collapse.

B Adjusted preprocessing strategy for DiffuRec

DiffuRec originally adopted a subsequence splitting strategy that may *unfairly* leverage more training data. For instance, given a sequence of length 200 with a maximum truncation length of 50, standard preprocessing keeps only the last 50 interactions. In contrast, DiffuRec splits all 200 interactions into training subsequences. *To ensure a fair comparison with other baselines, we revised its preprocessing pipeline: sequences are first truncated to the maximum length and only then split into subsequences.*

C Linear Probe

To verify whether the model has formed a structured embedding representation, we conducted a linear probing experiment on the ML-100K dataset. We first load the trained embedding weights, perform a batch normalization (BatchNorm1d) operation, and then feed them into a linear classification head. During training, the embedding weights are frozen. In the ML-100K dataset, each item has 26 category attributes: ['Action', 'Crime', 'Film-Noir', 'Musical', 'Sci-Fi', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Documentary', 'Drama', 'Family', 'Fantasy', 'Game-Show', 'History', 'Horror', 'Music', 'Mystery', 'News', 'Reality-TV', 'Romance', 'Short', 'Sport', 'Talk-Show', 'Thriller', 'War', 'Western']. We use a BCE loss function, with an Adam optimizer and a learning rate of 0.01, and train for 20 epochs.

D Positional Encoding

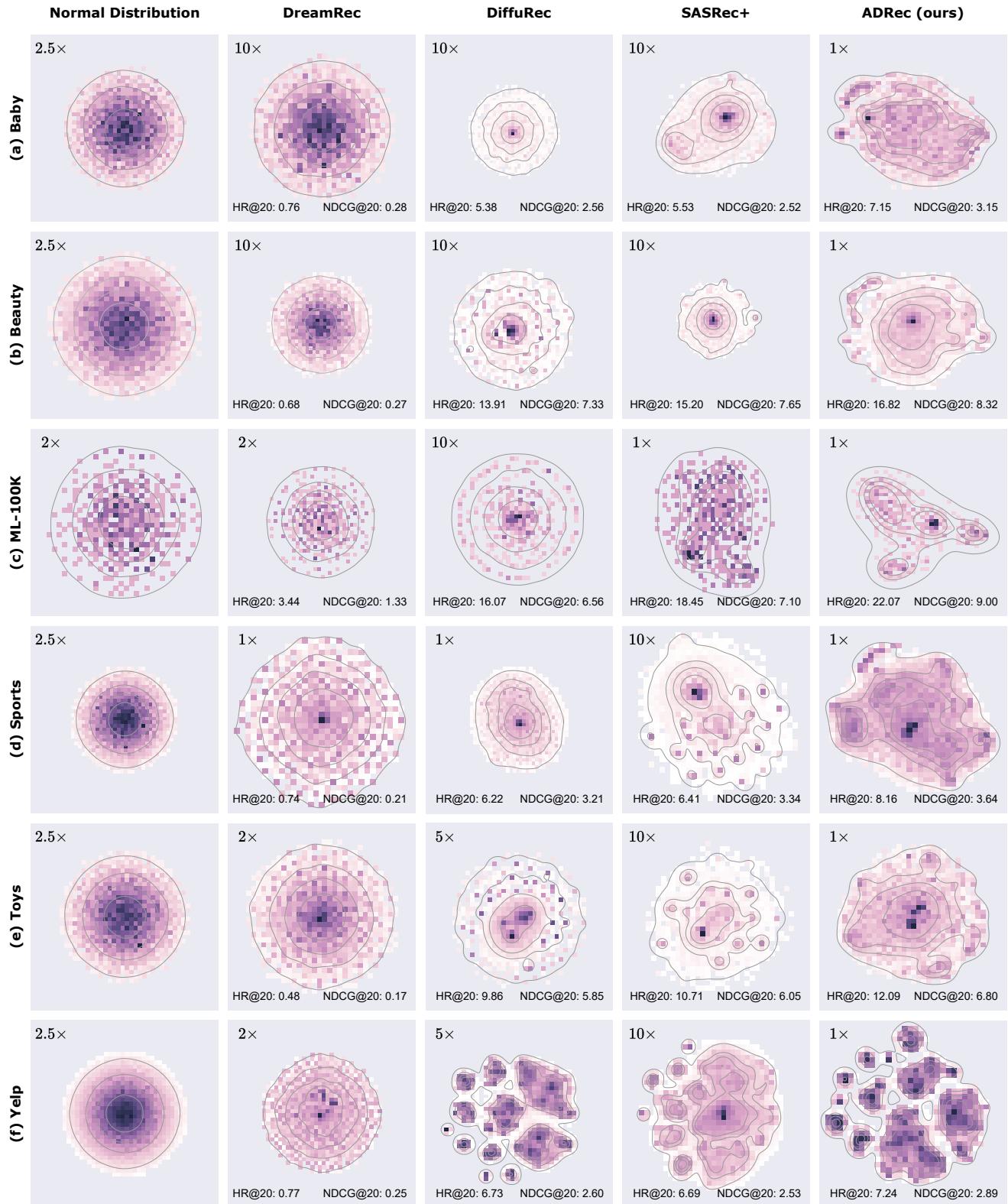
We find that ADRec does not require positional encoding. Adding positional encoding leads to a slight decrease in performance (Figure 8 in the Appendix). This is similar to the conclusion in [8], where it was observed that in some tasks, causal attention allows the model to approximate the absolute position of each token by inferring the number of preceding tokens it can attend to. This is closely related to the per-step prediction auto-regressive strategy we employ.

It becomes more complicated when introducing diffusion models. Potential insights based on our understanding are as follows:

- The noisy input at each diffusion step may already be disrupted, rendering the positional encoding information unstable or meaningless.
- Forcing positional encoding may lead the Transformer to learn “pseudo-patterns,” introducing additional bias and interfering with the true denoising process.
- Positional encoding also acts as meaningless noise for the diffusion model, deviating from the assumption that noise gradually converges to a Gaussian distribution.

E Diffusion Steps

In Appendix Figure 9, we demonstrate that ADRec is not sensitive to the diffusion time steps. This could be because the auto-regressive and diffusion models jointly drive the recommendation results. The discriminative power of the auto-regressive model over the structured embedding space defines the lower bound of ADRec’s performance, while a substantial embedding space relies on the distribution modeling capabilities of the diffusion model.

**Figure 10: t-SNE results on six datasets.**