

DiffuRec: A Diffusion Model for Sequential Recommendation

ZIHAO LI and CHENLIANG LI*, Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China
AIXIN SUN, Nanyang Technological University, Singapore

Mainstream solutions to Sequential Recommendation (SR) represent items with fixed vectors. These vectors have limited capability in capturing items' latent aspects and users' diverse preferences. As a new generative paradigm, *Diffusion models* have achieved excellent performance in areas like computer vision and natural language processing. To our understanding, its unique merit in representation generation well fits the problem setting of sequential recommendation. In this paper, we make the very first attempt to adapt Diffusion model to SR and propose DIFFUREC, for item representation construction and uncertainty injection. Rather than modeling item representations as fixed vectors, we represent them as distributions in DIFFUREC, which reflect user's multiple interests and item's various aspects adaptively. In diffusion phase, DIFFUREC corrupts the target item embedding into a Gaussian distribution via noise adding, which is further applied for sequential item distribution representation generation and uncertainty injection. Afterward, the item representation is fed into an Approximator for target item representation reconstruction. In reverse phase, based on user's historical interaction behaviors, we reverse a Gaussian noise into the target item representation, then apply a rounding operation for target item prediction. Experiments over four datasets show that DIFFUREC outperforms strong baselines by a large margin¹.

CCS Concepts: • **Information systems → Recommender systems.**

Additional Key Words and Phrases: Diffusion Model, Sequential Recommendation, User Preference Learning

ACM Reference Format:

Zihao Li, Chenliang Li, and Aixin Sun. 2023. DiffuRec: A Diffusion Model for Sequential Recommendation. *ACM Trans. Inf. Syst.* 37, 4, Article 111 (August 2023), 28 pages. <https://doi.org/XXXXXX.XXXXXXX>

1 INTRODUCTION

Sequential Recommendation (SR) aims to model a user's interest evolution process, based on his/her historical interaction records, for the next item prediction. The task has attracted widespread attention recently due to its significant business value. Seen as a sequence modeling task, sequential neural networks like LSTM have been widely adopted in early studies [18, 19]. Recent efforts illustrated the remarkable ability of self-attention mechanism for long-term dependency capturing. Thus, self-attention based methods became a prominent solution [25, 42, 50, 55]. SASRec [25] is a pioneering work that applies a uni-direction Transformer to learn transition patterns of items in sequences. BERT4Rec [42] adopted a bi-direction Transformer encoder to predict the next item.

*Chenliang Li is the corresponding author.

¹Code is available at: <https://github.com/WHUIR/DiffuRec>

Authors' addresses: Zihao Li, zihao.li@whu.edu.cn; Chenliang Li, cllee@whu.edu.cn, Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, 299 Bayi Road, Wuchang District, Wuhan, Hubei, China, 430072; Aixin Sun, Nanyang Technological University, Singapore, axsun@ntu.edu.sg.

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 ACM 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.

© 2023 Association for Computing Machinery.

1046-8188/2023/8-ART111 \$15.00

<https://doi.org/XXXXXX.XXXXXXX>



Fig. 1. An example of multiple interests of users and multiple aspects of items.

Based on information propagation and aggregation, Graph Neural Network (GNN) has also achieved impressive results on SR [11, 14]. Compared to earlier solutions, GNN could capture high-order dependencies between items.

All these mainstream methods learn item representation as an embedding vector. However, we believe a fixed vector may have limited capability in capturing the following four characteristics simultaneously.

- **Multiple Latent Aspects.** In practical scenarios, items may contain different latent aspects. We take the movie recommendation scenario as a running example (as shown in Figure 1). The aspects of items can be regarded as the themes or categories of movies. Thus, different movies will contain different theme distributions (circled on the left). For example, *Titanic* is a romantic movie, while *Life is Beautiful* contains both themes of Love and War. Note that in many recommendation scenarios, it is difficult to clearly define and precisely annotate items with their corresponding aspects. Encoding the complex latent aspects in a single vector remains challenging [9, 10].
- **Multiple Interests.** User's interests and preferences are diverse and inconstant. Further, user may dynamically adjust her interests based on different scenarios or periods. Hence, her focus on the various latent aspects of an item may also drift. As shown in Figure 1, when this user's preference to the themes (*i.e.*, Love and War) is changed. The importance of each latent aspect for an item should also be adjusted accordingly, in a user-aware manner. From this perspective, a static item representation across users is insufficient as well [10].
- **Uncertainty.** Due to the diversity and evolution of user interests, user's current preference and intention become uncertain to some degree. On the other hand, the diversity, novelty, serendipity, and uncertainty reflected through the recommended items are also expected from a recommender system [9, 24, 58]. We argue that it is more appropriate to model user's current interests as a distribution.

- **Guidance of Target Item.** At last, the target item could facilitate the user’s current intention understanding, and it can be introduced as an auxiliary signal to guide the latter process [48]. However, since the interaction involving the target item could introduce tremendous computation cost and deep coupling for the network design, the inference speed is also highly related to the size of the candidate pool. Although few works have been proposed, they are only applicable to the ranking stage.

To partially address the aforementioned characteristics, attention mechanism and dynamic routing for multi-interest modeling and recommendation have been explored [4, 27, 48, 60]. However, those methods require a proper number of interests (or capsules) to be heuristically pre-defined for model training. Besides, this pre-defined and fixed number also constrains the representation ability and flexibility of models. In addition to these solutions, Variational Auto-Encoder (VAE) methods have the potential to model the multiple latent aspects of an item as a distribution and inject uncertainty to the models via Gaussian noises. However, VAE suffers from the limited representation capacity [53] and collapse issue of posterior distribution approximation [29, 59], precluding the models to achieve the best performance. Furthermore, none of existing methods could model the multiple interests of users and multiple latent aspects of items as distributions, under a unified framework.

Diffusion models have made remarkable success in CV, NLP and many other fields [1, 28, 32, 54]. With the merits of its distribution generation and diversity representation, we consider Diffusion model to be a good fit to sequential recommendation. In this paper, we thereby make the very first attempt to bridge the gap between diffusion model and sequential recommendation, and propose DIFFUREC.

In the proposed DIFFUREC, in diffusion phase, we gradually add Gaussian noise into the target item embedding. The noised target item representation is then fed as input to generate distribution representations for the historical items, *i.e.*, to exploit the guidance of target item. As such, the item’s multi-latent aspects and user’s multi-interests could be represented as distributions instead of fixed vectors and the supervised signal encapsulated in the target item could also be fused for representation generation simultaneously. Note that the noised target item representation also injects some uncertainty into this process, leading to more discriminative item embedding learning. Then, based on the distribution representations of these historical items, we utilize a Transformer model as *Approximator*, to reconstruct target item representation for model training. As to the inference, we iteratively reverse a pure Gaussian noise into the target item distribution representation. The well-trained approximator will reveal the user’s current interest iteratively. Furthermore, considering the reversed target item representation distributes in a high-dimensional continuous space [28], we devise a rounding function to map it into a discrete item embedding space for target item prediction. In a nutshell, we make the following contributions in this paper:

- To the best of our knowledge, this is the first attempt to apply diffusion model for sequential recommendation. Thanks to the diffusion model’s inherent capability of distribution generation, we are able to model item’s multi-latent aspects and user’s diverse intentions as distributions.
- We devise a diffusion model, which can inherently fuse the target item embedding for historical interacted item representation generation. Thus, user’s current preference and intentions could be introduced as an auxiliary signal for better user understanding. Besides, some uncertainty could also be injected into modeling training process to enhance model robustness.
- We conduct experiments on four real-world datasets to verify the effectiveness of DIFFUREC. The results show that DIFFUREC outperforms nine existing strong baselines by a large margin.

Extensive ablation studies and empirical analysis also elucidate the effectiveness of our proposed components.

The rest of this paper is organized as follows: In Section 2, we will briefly introduce the relevant works to our method including some typical models for sequential recommendation, multi-interesting modeling, and representative methods specialized in distribution representation and uncertainty injection. In Section 4, the background knowledge about the diffusion model will be illustrated first. Then, we will present a whole pipeline of DIFFUREC so that the readers could obtain a general impression to our method. Subsequently, the key modules and the algorithm in this method, e.g., diffusion and reverse phases, the architecture of *Approximator*, will also be elucidated in detail. Section 5 will introduce the evaluated public datasets and metrics, experiments setting, and the overall performance of our method compared to other baselines. Additionally, some in-depth analysis e.g., training efficiency and convergence, ablation studies, performance on head and long-tail items, and uncertainty visualization, will also be given to exhibit the effectiveness of our method. Finally, we will conclude this work and discuss future research directions in Section 6.

2 RELATED WORK

We first briefly review the representative methods for sequential recommendation. We then review the related studies on multi-interest modeling and distribution representation generation, the two major areas that are most related to our work. Lastly, we introduce the development of diffusion models in different research areas.

2.1 Sequential Recommendation

Markov Chain played a crucial role in sequential recommendation in the early stage [33, 38, 62], for modeling a user's interaction behavior as a Markov Decision Process (MDP). Although Markov chains achieved remarkable success for sequential recommendation in the early years, the strong assumption that the next clicked item only relies on the previous one heavily confined the representation ability of this method. In deep learning era, many deep sequential neural networks, e.g., GRU, LSTM, CNN, and further variants [19, 52], emerged and delivered promising performance on this task. GRU4REC [19] is a pioneering work that stacks multiple GRU layers for sequential information modeling and applies a session-parallel mini-batches training strategy for next item recommendation. Caser [44] models recent K adjacent items as an "image" then apply convolution operation to capture the sequential patterns for next-item recommendation. Except that, other advancements of sequence modeling, e.g., attention mechanisms [25, 42, 44] and memory network [6, 43], also demonstrated their effectiveness with state-of-the-art performance in this area. SASRec [25] and BERT4Rec [42] are two representative works that apply self-attention for sequential recommendation. To be specific, SASRec regards the next-item prediction as an auto-regressive task and utilizes a uni-direction mask self-attention to capture the previous item information for next item prediction. In contrast, BERT4Rec argues that future information will also facilitate the current item predictions. Therefore, it models the sequential recommendation as a cloze task then applies bi-direction self-attention to introduce the future information for recommendation. Moreover, Chen *et al.* [6] devised a memory-augmented neural network (MANN) with collaborative filtering which contains item-level and future-level specifications for sequential recommendation. On top of that, Tan *et al.* [43] proposed a novel dynamic memory-based attention network (DMAN), which truncates long sequences into several sequence fragments and applies a memory network in a dynamic fashion to extract users' both short-term and long-term interests for recommendation.

Attributed to information propagation and aggregation, graph neural network is more effective in incorporating high-order relationships in a sequence. Hence it was applied for sequential recommendation [8, 11, 14]. Wu *et al.* [51] proposed SR-GNN, which is the first exploration that introduces a graph neural network to model explicit relationships between items for next-item recommendation. Additionally, Fan *et al.* [11] elaborated to model time interval information and collaborate filter signal with graph neural network for sequential recommendation. Ding *et al.* [8] constructed the user-item interaction graph and the item-item transition graph to extract more complicated collaborative filtering signals and item transitional patterns for user and item embedding enhancement and sequential recommendation.

2.2 Multi-Interest Modeling

As users' interests are dynamic and diverse, multi-interest modeling with soft-attention mechanism and dynamic routing became prominent for sequential recommendation. For example, Zhuo *et al.* [60] believes utilizing a single fixed vector can not fully express the user's diversity preference, thus, a deep interest network (DIN) with a local activation unit was designed to adaptively learn user's interest representation from his/her historical interaction sequence. As such, the expressive capability of the model could be enhanced to fully express users' diverse interests. Besides, MIND [27] designed a multi-interest extractor layer via the capsule dynamic routing mechanism for user's multi-interest extraction and sequence recommendation. On top of that, Cen *et al.* further proposed ComiRec, which leverages two modules *i.e.*, dynamic routing and attention mechanism respectively for multi-interests modeling and recommendation. More recent works, *e.g.*, Re4 [56], TiMiRec [48] strive to introduce auxiliary loss functions for target interest distillation and distinction. MGNM [45] combines graph convolution operation with dynamic routing for more precisely multi-level interest learning. Considering user's preference evolution process may be uncertain and volatile, SPLIT [39] applies reinforcement learning to decompose independent preference from user's historical behavior sequence for multi-interest modeling and sequential recommendation. However, all these solutions require to pre-define a proper number of interests. This process may be heuristic and time-consuming. Additionally, the fixed interest number also confines the model from learning complicated transition patterns in a flexible manner, leading the model to achieve sub-optimal performance.

2.3 Representation Distribution and Uncertainty Modeling

As aforementioned, user's preference is uncertain and diverse in practical scenarios, thus, a unified representation vector is insufficient to model user's dynamic preference in an adaptive manner. VAE, with its ability to model probabilistic latent variables, has been used for generating representation distributions and injecting uncertainty in sequential recommendation. For example, Liang *et al.* proposed Multi-VAE [29], the pioneering work that applied VAE to collaborative filtering for recommendation. Sachdeva *et al.* combined recurrent neural network with VAE to capture temporal patterns for sequential recommendation [36]. Considering the posterior distribution approximation hurts the representation ability of conventional VAE methods, ACVAE introduced the Adversarial variational Bayes (AVB) framework to enhance the representation of latent variables [53]. Wang *et al.* further incorporated contrastive learning into the VAE paradigm to alleviate the representation degeneration problem [49]. Different from these VAE methods, Fan *et al.* [9] proposed a Distribution-based Transform, which models the item representation as Elliptical Gaussian distributions where the conventional unified item embedding is served as means to reveal user's basic interests, besides a stochastic vector is also introduced for covariance representation and uncertainty injection. Moreover, the authors utilized the Wasserstein Distance as a loss function to optimize the distance between sequential distribution representation and item representation

distribution. On that basis, Fan *et al.* [10] further proposed a novel stochastic self-attention (STOSA) model very recently, which replaces the inner product between any of two items in self-attention module as Wasserstein Distance for correlation measuring and collaborative transitivity capture. Nevertheless, VAE methods have the ability to model latent variables as distributions, they struggle with representation degeneration and collapse issues.

2.4 Diffusion Models

Motivated by non-equilibrium thermodynamics [40], Diffusion model have shown its great potential in computer vision [3, 21, 22, 32], natural language processing [12, 17, 28] and other areas [5]. For instance, Super-Resolution via Repeated Refinement (SR3) [37] models the super-resolution task as a stochastic, iterative denoising process. Then utilizing Denoising Diffusion Probabilistic Models (DDPM) for conditional image generation. DALLE-2 [32] applies a two-stage modeling method for text-conditional image generation. To be specific, it first trains a text encoder and image encoder with CLIP strategy for text representation and image representation alignment. Then, given any text description, we could obtain the representation via the well-trained text encoder, which will be further fed into an autoregressive or diffusion prior for image representation generation. Conditioned on this image representation, a diffusion-based decoder is applied to produce the final image. In the NLP area, Diffusion-LM [28] is the first effort that adapted continuous diffusion to instantiate the idea of fine-grained control on NLP-oriented tasks. Following Diffusion-LM, DiffuSeq made the extension to support more general sequence-to-sequence tasks [12]. Specifically, in training or diffusion process, given source and target text pairs, DiffuSeq first corrupts the target text sequence into Gaussian noises. Then, based on the source text sequence and approximator model to reconstruct the target text sequence. In reverse or inference process, given any pure Gaussian noise and source text, we could reverse the target text iteratively from the noise via the well-trained approximator. Although Diffusion models became ubiquitous in other domains, adapting the model to recommender systems remains under-explored. We believe Diffusion model is a good fit to SR, and make the first attempt in this paper.

3 PRELIMINARY

Before presenting our model, we first briefly introduce the standard diffusion model as preliminary knowledge. Endowed with the property of latent variable modeling, diffusion model could generate continuous diversity distribution representation in the diffusion and reverse phases.

In **diffusion phase**, the diffusion model incrementally corrupts the original representation \mathbf{x}_0 into a Gaussian noise \mathbf{x}_t via a Markov Chain (*i.e.*, $\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \dots \rightarrow \mathbf{x}_{t-1} \rightarrow \mathbf{x}_t$). This process can be formalized as follows:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N} \left(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I} \right) \quad (1)$$

where $\mathcal{N}(\mathbf{x}; \mu, \sigma^2)$ is a Gaussian distribution with a mean $\mu = \sqrt{1 - \beta_t}$ and variance $\sigma^2 = \beta_t$, \mathbf{x}_t is sampled from this Gaussian distribution, β_t controls the noise added at the t -th diffusion step and \mathbf{I} is an identity matrix.

The value of β_t is generated from a pre-defined noise schedule β which arranges how much noise is injected at which step. The common noise schedule includes square-root [28], cosine [20], and linear [31]. According to [20], at an arbitrary diffusion step t , we can derive \mathbf{x}_t conditioned on the input \mathbf{x}_0 in a straightforward way following the Markov chain process. Then Equation 1 can be rewritten as follows:

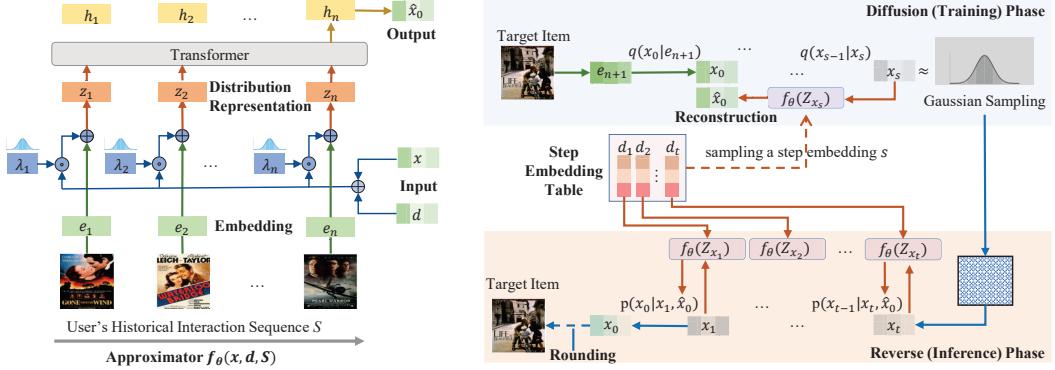


Fig. 2. Architecture of DIFFUREC. The figure on the left is the *Approximator*, a Transformer backbone for target item representation reconstruction. The two figures on the right illustrate the diffusion phase and the reverse phase, respectively.

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N} \left(\mathbf{x}_t; \sqrt{\alpha_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I} \right) \quad (2)$$

$$\bar{\alpha}_t = \prod_{s=1}^t \alpha_s, \quad \alpha_s = 1 - \beta_s \quad (3)$$

In **reverse phase**, a standard Gaussian representation \mathbf{x}_t is denoised to approximate the real representation \mathbf{x}_0 (*i.e.*, $\mathbf{x}_t \rightarrow \mathbf{x}_{t-1} \rightarrow \dots, \mathbf{x}_1, \rightarrow, \mathbf{x}_0$) in an iterative manner. Specifically, given the current denoised representation \mathbf{x}_s , the next representation \mathbf{x}_{s-1} after one-step reverse is calculated as follows:

$$p(\mathbf{x}_{s-1} | \mathbf{x}_s, \mathbf{x}_0) = \mathcal{N} \left(\mathbf{x}_{s-1}; \tilde{\mu}_s(\mathbf{x}_s, \mathbf{x}_0), \tilde{\beta}_s \mathbf{I} \right) \quad (4)$$

$$\tilde{\mu}_s(\mathbf{x}_s, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{s-1}} \beta_s}{1 - \bar{\alpha}_s} \mathbf{x}_0 + \frac{\sqrt{\alpha_s} (1 - \bar{\alpha}_{s-1})}{1 - \bar{\alpha}_s} \mathbf{x}_s \quad (5)$$

$$\tilde{\beta}_s = \frac{1 - \bar{\alpha}_{s-1}}{1 - \bar{\alpha}_s} \beta_s \quad (6)$$

However, in the reverse phase, the \mathbf{x}_0 is unknown, a deep neural network $f_\theta(\cdot)$ (*e.g.*, Transformer [47] or U-Net [35]) is generally used for estimating \mathbf{x}_0 . Given the original representation \mathbf{x}_0 and the schedule β , $f_\theta(\cdot)$ is trained to minimize the following variational lower bound (VLB) [40]. Below is the derivation of the VLB, we include it here for completeness:

$$\begin{aligned} \mathcal{L}_{vlb} &= \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:t})}{q(\mathbf{x}_{1:t} | \mathbf{x}_0)} \right] \\ &= \mathbb{E}_q \left[-\log p(\mathbf{x}_t) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_t | \mathbf{x}_{t-1})} \right] \\ &= \mathbb{E}_q \left[-\log p(\mathbf{x}_t) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_t | \mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}{q(\mathbf{x}_1 | \mathbf{x}_0)} \right] \end{aligned} \quad (7)$$

$$\begin{aligned}
&= \mathbb{E}_q \left[-\log p(\mathbf{x}_t) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} \right] \\
&= \mathbb{E}_q \left[-\log \frac{p(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \right] \\
&= \mathbb{E}_q \left[\underbrace{D_{KL}(q(\mathbf{x}_t|\mathbf{x}_0)||p(\mathbf{x}_t))}_{L_t} + \underbrace{\sum_{t>1} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right]
\end{aligned}$$

where $p(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \mathbf{0}, \mathbf{I})$, and $D_{KL}(\cdot)$ is KL divergence measure.

Here, we organize this loss (ref. Equation 7) into three parts: L_t , L_{t-1} and L_0 . Part L_t works to push the $q(\mathbf{x}_t|\mathbf{x}_0)$ close to a standard Gaussian distribution. Part L_{t-1} works to minimize the KL divergence between the forward process posterior and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ (generated by the deep neural network) for reverse process. The last part L_0 utilizes negative log-likelihood for final prediction. Note that this VLB loss could easily lead to unstable model training. To alleviate this issue, following [20], we could expand and reweight each KL-divergence term in VLB with a specific parameterization to obtain a simple mean-squared error loss as follows:

$$\mathcal{L}_{simple} = \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2 \right] \quad (8)$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is sampled from a standard Gaussian distribution for noise injection and diffusion, and $\epsilon_\theta(\cdot)$ works as a **Approximator** that can be instantiated by a deep neural network e.g., Transformer or U-Net.

4 METHODOLOGY

We first formally define the task of sequential recommendation, then we introduce the technical detail of our proposed DIFFUREC model including diffusion and revers process for model training and inference respectively, approximator for sequence modeling and target item representation reconstruction, rounding operation for target item index projection, and loss function for model optimization.

Sequential Recommendation. Given a set of items \mathcal{I} , a set of users \mathcal{U} , and their interaction records, we can organize each user's interactions chronologically as a sequence $\mathcal{S} = [i_1^u, i_2^u, \dots, i_{n-1}^u, i_n^u]$. Here, $u \in \mathcal{U}$, $i_j^u \in \mathcal{I}$ denotes the j -th item interacted with user u . n is the sequence length and is also the total number of interactions from user u . The goal of sequential recommendation is to generate a ranked list of items as the predicted candidates for next item that will be preferred by the user u .

As its name suggests, DIFFUREC is built upon the diffusion model. The overall architecture of DIFFUREC is shown in Figure 2, with three major parts: 1) Approximator for target item representation reconstruction; 2) Diffusion (Training) process to incorporate the guidance of target item and inject noise for robust approximator learning; and 3) Reverse (Inference) phase for target item prediction.

4.1 Overview

Refer to Figure 2, at first, we consider a static item embedding \mathbf{e}_j as the semantic encoding of the intrinsic latent aspects covered by item i_j . Following the diffusion model described in Section 3,

we plan to perform reverse to recover the target item from the historical sequence \mathcal{S} . Hence, we add noise into target item embedding \mathbf{e}_{n+1} through the diffusion process. Recall that the diffusion process involves a series of samplings from Gaussian distributions (ref. Equation 1). We treat the noised target item representation \mathbf{x}_s undergo s -diffusion steps as the distribution representation sampled from $q(\cdot)$. Similar to adversarial learning [13], the injected noise can help us to derive sharper item embeddings, thus \mathbf{x}_s can still reflect the information of multiple latent aspects of the item. Then, we use \mathbf{x}_s to adjust the representation of each historical item in \mathcal{S} , such that the guidance of target item is exploited as auxiliary semantic signals. Afterwards, the resultant representations $\mathbf{Z}_{\mathbf{x}_s} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n]$ are fed into approximator $f_\theta(\cdot)$. That is, the model training is performed to reinforce the reconstructed $\hat{\mathbf{x}}_0$ from the approximator close to target item embedding \mathbf{e}_{n+1} .

As to the reverse phase, we firstly sample the noised target item representation \mathbf{x}_t from a standard Gaussian distribution, *i.e.*, $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Similar to the diffusion process, the resultant representations $\mathbf{Z}_{\mathbf{x}_t}$ adjusted by using \mathbf{x}_t are fed into the well-trained approximator $f_\theta(\cdot)$ for $\hat{\mathbf{x}}_0$ estimate. Following Equation 4, the estimated $\hat{\mathbf{x}}_0$ and \mathbf{x}_t are used to reverse \mathbf{x}_{t-1} via $p(\cdot)$. This process continues iteratively until \mathbf{x}_0 is arrived. Note that the reverse phase is also stochastic, reaching the purpose of modeling uncertainty in user behavior. Then we utilize a rounding function to map the reversed continuous representation \mathbf{x}_0 into discrete candidate item indexes for target item \hat{i}_{n+1} prediction.

Formally, the above diffusion, reverse process and rounding operation can be formulated as follows:

- **Diffusion:**

$$\begin{aligned}\mathbf{x}_s &\leftarrow q(\mathbf{x}_s | \mathbf{x}_0, s) \\ \hat{\mathbf{x}}_0 &= f_\theta(\mathbf{Z}_{\mathbf{x}_s})\end{aligned}\tag{9}$$

- **Reverse:**

$$\begin{aligned}\hat{\mathbf{x}}_0 &= f_\theta(\mathbf{Z}_{\mathbf{x}_t}) \\ \mathbf{x}_{t-1} &\leftarrow p(\mathbf{x}_{t-1} | \hat{\mathbf{x}}_0, \mathbf{x}_t)\end{aligned}\tag{10}$$

- **Rounding:**

$$\hat{i}_{n+1} \leftarrow \text{Rounding}(\mathbf{x}_0)\tag{11}$$

4.2 Approximator

Instead of using U-Net in CV field [32, 34], following [12, 28], we utilize Transformer as the backbone of Approximator $f_\theta(\cdot)$ to generate $\hat{\mathbf{x}}_0$ in diffusion and reverse phases, since Transformer has been well proven to be effective for sequential dependencies modeling and it is ubiquitous in NLP and recommendation tasks.

$$\hat{\mathbf{x}}_0 = f_\theta(\mathbf{Z}_{\mathbf{x}}) = \text{Transformer}([\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_n])\tag{12}$$

$$\mathbf{z}_i = \mathbf{e}_i + \lambda_i \odot (\mathbf{x} + \mathbf{d})\tag{13}$$

where symbol \odot is the element-wise product, and \mathbf{d} is the embedding of the corresponding diffusion/reverse step created by following [47]. Through step embeddings, the specific information of each diffusion and reverse step could be introduced into the model. As discussed earlier, in diffusion phase, \mathbf{x} is the noised target item embedding \mathbf{x}_s (more discussion in Section 4.3). And in reverse process, \mathbf{x} is the reversed target item representation, *i.e.*, $\mathbf{x} = \mathbf{x}_s$ ($s = t, t-1, \dots, 2, 1$). In Equation 13, λ_i is sampled from a Gaussian distribution, *i.e.*, $\lambda_i \sim \mathcal{N}(\delta, \delta)$, δ is a hyperparameter which defines both the mean and variance. For diffusion phase, λ_i controls the extent of noise injection. On the

other hand, this setting introduces uncertainty to model the user's interest evolution for reverse phase. That is, the importance of each latent aspect of a historical item can be adjusted iteratively along the reverse steps in a user-aware manner.

We keep the same as the standard Transformer architecture and configuration [47], *i.e.*, the multi-head self-attention, feed-forward network with ReLU activation function, layer normalization, dropout strategy, and residual connection are utilized. To enhance the representation ability of models, we also stack multi-blocks Transformer. Finally, we gather the representation \mathbf{h}_n of item i_n^u generated by the last layer as $\hat{\mathbf{x}}_0$.

4.3 Diffusion Phase

The key part of the diffusion phase is to design a noise schedule β_s to arrange the noise injection in step s for uncertainty modeling. Following [23, 40], a uniform noise schedule was explored for the discrete domain. In this paper, we utilize a truncated linear schedule for β_s generation (ref. Figure 3 and Figure 4 for more detailed discussion) which can be formalized as follows:

$$\beta_s = \begin{cases} \frac{a}{t}s + \frac{b}{s}, & \beta_s \leq \tau \\ \frac{1}{10}\left(\frac{a}{t}s + \frac{b}{s}\right), & \text{others} \end{cases} \quad (14)$$

where a, b are hyperparameters, which control the range of β . If $\beta_s > \tau$, we define $\beta_s = 0.1\beta_s$. τ is a truncated threshold, we set $\tau = 1$ in our paper. Denoting the maximum diffusion step as t , in the training process, we randomly sample a diffusion step s for each target item, *i.e.*, $s = \lfloor s' \rfloor$, $s' \sim U(0, t)$. Following Equation 9, we can generate \mathbf{x}_s via $q(\mathbf{x}_s | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_s; \sqrt{\alpha_t} \mathbf{x}_0, (1 - \bar{\alpha}_s) \mathbf{I})$. We choose to derive \mathbf{x}_0 with one step diffusion from the target item embedding: *i.e.*, $q(\mathbf{x}_0 | \mathbf{e}_{n+1}) = \mathcal{N}(\mathbf{x}_0; \sqrt{\alpha_0} \mathbf{e}_{n+1}, (1 - \alpha_0) \mathbf{I})$. Actually, based on the reparameter trick (*i.e.*, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$), we could generate \mathbf{x}_s as follows:

$$\mathbf{x}_s = \sqrt{\alpha_s} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_s} \epsilon \quad (15)$$

4.4 Reverse Phase

As aforementioned, we aim to recover the target item representation \mathbf{x}_0 iteratively from a pure Gaussian noise \mathbf{x}_t , in the reverse phase. However, this process will be intractable as \mathbf{x}_0 is required at each reverse step. Consequently, we use the well-trained approximator to generate $\hat{\mathbf{x}}_0$ for \mathbf{x}_0 estimation, *i.e.*, $\mathbf{x}_0 = \hat{\mathbf{x}}_0$. Following Equation 4, the reverse step is performed after applying reparameter trick:

$$\hat{\mathbf{x}}_0 = f_\theta(\mathbf{Z}_{x_t}) \quad (16)$$

$$\mathbf{x}_{t-1} = \tilde{\mu}_t(\mathbf{x}_t, \hat{\mathbf{x}}_0) + \tilde{\beta}_t \epsilon' \quad (17)$$

where $\tilde{\mu}_t(\mathbf{x}_t, \hat{\mathbf{x}}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} \hat{\mathbf{x}}_0 + \frac{\sqrt{\alpha_t} (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t$ and $\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$, and $\epsilon' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for \mathbf{x}_{t-1} generation. Repeat the above process until we arrive at \mathbf{x}_0 . The reverse phase can be illustrated in Algorithm 1.

4.5 Loss Function and Rounding

As the approximator requires the step embedding for reconstruction (ref. Equation 13), in diffusion phase, we randomly sample a step-index s from a uniform distribution over range $[1, t]$. So as to the reverse phase, the step-index will be from t to 1, where t is the total steps. Therefore, the corresponding step embedding and sequence item distribution are fed into the approximator for model learning.

Algorithm 1: Reverse or Inference

```

1: Input:
2:   Historical Sequence:  $(i_1, \dots, i_n)$ ;
3:   Target item Representation  $\mathbf{x}_t$ :  $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ;
4:   Total Reverse Steps:  $t$ ;
5:   Schedule  $\beta$ : Linear;
6:   Hyperparameter  $\delta$  (mean and variance):  $\lambda$  sampling;
7:   Approximator:  $f_\theta(\mathbf{x})$ ;
8: Output:
9:   Predicted Target Item:  $i_{n+1}$ ;
10:   $s = t$ 
11:  while  $s > 0$  do
12:     $(\mathbf{z}_1, \dots, \mathbf{z}_n) \leftarrow (\mathbf{e}_1 + \lambda_1(\mathbf{x}_s + \mathbf{d}_s), \dots, \mathbf{e}_n + \lambda_n(\mathbf{x}_s + \mathbf{d}_s))$ ,  $\lambda_i \sim \mathcal{N}(\delta, \delta)$ ; // Distribution
        Representation
13:     $\hat{\mathbf{x}}_0 \leftarrow f_\theta(\mathbf{z}_1, \dots, \mathbf{z}_n)$ ; // Reconstruction
14:     $\mathbf{x}_{s-1} \leftarrow \tilde{\mu}(\hat{\mathbf{x}}_0, \mathbf{x}_s) + \tilde{\beta}_t * \epsilon'$ , ( $\epsilon' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ); // reverse
15:     $s = s - 1$ ; // Iteration
16:  end while
17:   $i_{n+1} \leftarrow \text{Rounding}(\mathbf{x}_0)$ ;

```

Algorithm 2: Diffusion or Training

```

1: Input:
2:   Historical Sequence:  $(i_1, \dots, i_n, i_{n+1})$ ;
3:   Learning Epochs:  $E$ ;
4:   Maximum Diffusion Steps:  $t$ ;
5:   Schedule  $\beta$ : Linear;
6:   Hyperparameter  $\delta$  (mean and variance):  $\lambda$  sampling;
7:   Approximator:  $f_\theta(\cdot)$ ;
8: Output:
9:   Well-trained Approximator:  $f_\theta(\cdot)$ ;
10:  while  $j < E$  do
11:     $s = \lfloor s' \rfloor$ ,  $s' \sim U(0, t)$ ; // Diffusion Step Sampling
12:     $\mathbf{x}_s \leftarrow q(\mathbf{x}_s | \mathbf{x}_0)$ ; // Diffusion
13:     $(\mathbf{z}_1, \dots, \mathbf{z}_n) \leftarrow (\mathbf{e}_1 + \lambda_1(\mathbf{x}_s + \mathbf{d}_s), \dots, \mathbf{e}_n + \lambda_n(\mathbf{x}_s + \mathbf{d}_s))$ ,  $\lambda_i \sim \mathcal{N}(\delta, \delta)$ ; // Distribution
        Representation
14:     $\hat{\mathbf{x}}_0 \leftarrow f_\theta([\mathbf{z}_1, \dots, \mathbf{z}_n])$ ; // Reconstruction
15:    parameter update:  $\mathcal{L}_{CE}(\hat{\mathbf{x}}_0, i_{n+1})$ ;
16:     $j = j + 1$ ;
17:  end while

```

The standard objective function of diffusion models could be simplified as a mean-squared error (ref. Equation 8). However, we argue it may be inappropriate in recommendation tasks for two reasons. First, same as NLP, rather than a continuous distribution, the item embedding is static in

the latent space. Thus, the mean square error loss is known to be unstable in such a scenario [7, 30]. Second, it is more ubiquitous to calculate the relevance between two vectors in terms of inner product for sequential recommendation. Consequently, following [15, 30, 41], in diffusion phase, we instead utilize cross-entropy loss for model optimization as follows:

$$\hat{y} = \frac{\exp(\hat{\mathbf{x}}_0 \cdot \mathbf{e}_{n+1})}{\sum_{i \in \mathcal{I}} \exp(\hat{\mathbf{x}}_0 \cdot \mathbf{e}_i)} \quad (18)$$

$$\mathcal{L}_{CE} = \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} -\log \hat{y}_i \quad (19)$$

where $\hat{\mathbf{x}}_0$ is reconstructed by the Transformer based approximator and symbol \cdot indicates the inner product operation.

As for the inference phase, we need to map the reversed target item representation \mathbf{x}_0 into the discrete item index space for final recommendation. Here, we calculate the inner product between the \mathbf{x}_0 and all candidate item embeddings \mathbf{e}_i . And then, take the item index corresponding to the maximal as the final recommendation result. This can be formalized as follows:

$$\arg \max_{i \in \mathcal{I}} \text{Rounding}(\mathbf{x}_0) = \mathbf{x}_0 \cdot \mathbf{e}_i^T \quad (20)$$

Further, we also utilize inner product to rank the candidate items. The diffusion process or model training is in Algorithm 2.

4.6 Discussion

It is noteworthy that attributed to the property of diffusion model the corrupt or reversed target item representation \mathbf{x} will be some uncertainty in diffusion and reverse process. Additionally, we randomly sample a λ from Gaussian distribution for item distribution representation generation, this operation will also introduce some stochasticity and uncertainty in model training. We believe the injected uncertainty, *i.e.*, the small perturbation to the raw item embedding, will improve the robustness of model and also alleviate the over-fitting problem. To in-depth analyze the effects of this perturbation on the model final performance, following [13, 16], we implement an adversarial training framework, by adding adversarial perturbations on the embedding vectors for more robust model training and compare the performance of this strategy to our DIFFUREC.

More concretely, we suppose the adversarial perturbations to item embeddings that cause the dramatic change to the final results (a much worse recommendation performance) will be the effective perturbations. Hence, the model is not that robust and is vulnerable to such perturbations. We, thereby, apply adversarial training to improve the model's robustness to those perturbations. To be specific, the process of adversarial training can be transformed into a min-max optimization problem. First, by maximizing the loss function, we aim to find effective perturbations Δ . Afterward, the model is trained to minimize both the original loss $\mathcal{L}(\mathcal{D}|\Theta)$ and the additional loss with adversarial perturbations $\mathcal{L}(\mathcal{D}|\Theta + \Delta)$. The min-max optimization can be formalized as follows:

$$\Theta^*, \Delta^* = \arg \min_{\Theta} \max_{\Delta, \|\Delta\| \leq \epsilon} \mathcal{L}(\mathcal{D}|\Theta) + \gamma \mathcal{L}(\mathcal{D}|\Theta + \Delta) \quad (21)$$

where ϵ regulates the magnitude of the perturbations, γ balances the adversarial regularizer and main loss, *i.e.*, the cross-entropy loss for target item prediction. $\|\cdot\|$ denotes the L_2 norm. Θ is the learnable parameters of model.

However, it is intractable to deliver the exact value $\Delta_{adv} = \arg \max_{\Delta, \|\Delta\| \leq \epsilon} \mathcal{L}(\mathcal{D}|\Theta + \Delta)$ from Equation 21. As we add the perturbation in a linear manner for adversarial samples construction, following [13], we could move towards the gradient direction of the objective function with respect

to Δ when model training to approximate Δ_{adv} incrementally. This process can be formalized as follows:

$$\Delta_{adv} = \epsilon \frac{\Gamma}{\|\Gamma\|} \quad \text{where} \quad \Gamma = \frac{\partial \mathcal{L}(\mathcal{D}|\Theta + \Delta^t)}{\partial \Delta^t} \quad (22)$$

Denotes Δ^t as the perturbations in t -th training epoch. We initialize Δ^0 as zeros, and define $\Delta^{t+1} = \Delta_{adv}$.

Hence, the min-max optimization function can be rewritten as follows:

$$\mathcal{L}^*(\mathcal{D}|\Theta) = \mathcal{L}(\mathcal{D}|\Theta) + \gamma \mathcal{L}(\mathcal{D}|\Theta + \Delta_{adv}) \quad (23)$$

We use Equation 23 as the final loss function for adversarial training. For a fair comparison, we also apply the standard Transformer as a backbone and obtain the last hidden representation \mathbf{h}_n with regard to item i_n as the sequence representation and calculate its inner product with candidate item embeddings to yield the prediction scores for the next-item recommendation. Besides, the other setting, *i.e.*, item embedding size, hidden state dimensions, the number of transformer blocks and multi-heads, dropout ratio, etc, keep the same as DIFFUREC. The detailed experiment results analysis is presented in Section 5.7.

5 EXPERIMENTS

In this section, we select nine baselines covering three types of mainstream methods in sequential recommendation task, *e.g.*, representative sequential recommendation methods, multi-interesting modeling methods, VAE and uncertainty models. We conduct extensive experiments over four real-world datasets for performance evaluation. In detail, we aim to answer the following research questions:

- **RQ1.** How does the DIFFUREC perform compared with state-of-the-art sequential recommendation methods?
- **RQ2.** How does each design choice made in DIFFUREC affect its performance?
- **RQ3.** How do different hyperparameter settings affect its performance?
- **RQ4.** How does the DIFFUREC perform compared with adversarial training?
- **RQ5.** How does DIFFUREC perform on sequences of varying lengths and items of different popularity?
- **RQ6.** How efficient (*i.e.*, time complexity) is the training and inference of our proposed model compared with other methods?
- **RQ7.** How about the diversity and uncertainty of the final recommendation results of DIFFUREC?

5.1 Datasets and Evaluation Metrics

We use four real-world datasets to validate the efficacy of our DIFFUREC. All the datasets have been widely used for sequential recommendation.

- *Amazon Beauty* and *Amazon Toys*² are two categories of Amazon review datasets , which contains a collection of user-item interactions on Amazon spanning from May 1996 to July 2014.
- *Movielens-1M*³ is a widely-used benchmark dataset that includes one million movie ratings from 6000 users on 4000 movies.

²https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

³<http://files.grouplens.org/datasets/movielens/ml-1m.zip>

Table 1. Statistics of datasets after preprocessing. Avg_len means the average length of sequences.

Dataset	# Sequence	# items	# Actions	Avg_len	Sparsity
Beauty	22,363	12,101	198,502	8.53	99.93%
Toys	19,412	11,924	167,597	8.63	99.93%
Movielens-1M	6,040	3,416	999,611	165.50	95.16%
Steam	281,428	13,044	3,485,022	12.40	99.90%

- *Steam*⁴ is collected from a large online video game distribution platform. The dataset contains more than 40k games with abundant external information (spanning from October 2010 to January 2018), e.g., users' play hours, price, category, media score, publisher and developer information.

Following the common data preprocessing method [25, 42, 48], we treat all reviews or ratings as implicit feedback (*i.e.*, a user-item interaction) and chronologically organize them by their timestamps. Additionally, we filter out unpopular items and inactive users with fewer than 5 related actions. Moreover, we adopt a leave-one-out strategy for performance evaluation. To be specific, for all the datasets, given a sequence $\mathcal{S} = \{i_1, i_2, \dots, i_n\}$, we utilize the most recent interaction (i_n) for testing, the penultimate interaction (i_{n-1}) for model validation, and the earlier ones ($\{i_1, i_2, \dots, i_{n-2}\}$) for model training. The maximum sequence length is set to 200 for *MovieLens-1M* dataset, and 50 for the other three datasets. The statistics of these datasets are reported in Table 1. We could find that the average length of sequences and the size of these datasets are very different, covering a broad spectrum of real-world scenarios.

As for the evaluation metric, we evaluate all models with HR@K (Hit Rate) and NDCG@K (Normalized Discounted Cumulative Gain). We report the experimental results with $K = \{5, 10, 20\}$. HR@K represents the proportion of the hits recommended among the top-K list. NDCG@K further evaluates the ranking performance by considering the ranking positions of these hits. The NDCG@K is set to 0 when the rank exceeds K. Because the evaluation with sampling may cause inconsistency when the number of negative items is small [26], we rank all candidate items for target item prediction.

5.2 Baselines

We evaluate DIFFUREC against three types of representative sequential recommendation methods, including four conventional SR models, two models designed for multi-interest modeling, and three models dedicated to uncertainty modeling.

The Conventional Sequential Neural Models

- **GRU4Rec**⁵ [19] utilizes GRU to model the sequential behavior of users for recommendation.
- **Caser**⁶ [44] devises horizontal and vertical CNN to exploit user's recent sub-sequence behaviors for recommendation.
- **SASRec**⁷ [25] utilizes a single-direction Transformer with a mask encoder to model the implicit correlations between items and it is a competitive benchmark in sequential recommendation task.

⁴<https://steam.internet.byu.edu/>

⁵<https://github.com/hidasib/GRU4Rec>

⁶<https://github.com/graytowne/caser>

⁷<https://github.com/kang205/SASRec>

- **BERT4Rec**⁸ [42] believes the uni-directional architecture (e.g., SASRec) is insufficient for users' behaviors modeling. Consequently, a bidirectional Transformer with cloze task is proposed for sequential recommendation.

Multi-Interest Models

- **ComiRec**⁹ [4] adopts attention mechanism and dynamic routing for user's multi-interest extraction and recommendation.
- **TiMiRec**¹⁰ [48] is the most up-to-date multi-interest model. Compared with ComiRec, it designs an auxiliary loss function which involves the target item as a supervised signal for interest distribution generation.

VAE and Uncertainty Models

- **SVAE**¹¹ [36] is a pioneer work that combines GRU and variational autoencoder for next item prediction.
- **ACVAE**¹² [53] utilizes the Adversarial Variational Bayes (AVB) framework and propose an Adversarial and Contrastive Variational Autoencoder to generate high-quality latent variable representations for sequential recommendation.
- **STOSA**¹³ [10] develops the vanilla self-attention as a Wasserstein self-attention to model the inner correlation between any of two item representations and also inject some uncertainty into the model training process via a stochastic Gaussian distribution for sequential recommendation.

5.3 Implementation Details

For a fair comparison, we followed [25, 42, 61] and utilized the Adam optimizer with the initial learning rate of 0.001. We initialized the parameters of Transformer based on a Xavier normalization distribution and set the block numbers as 4. We fixed both the embedding dimension and all the hidden state size as 128 and batch size as 1024. The dropout rate of turning off neurons in Transformer block and item embedding are 0.1 and 0.3 for all datasets. So as to λ , we sample these values from a Gaussian distribution with both mean and variance of 0.001. The total number of reverse steps t is 32 and we further analyze the efficiency of reverse phase within the scope of {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024} (more discussion in Section 5.10). As for the noise schedule β , we select *truncated linear* schedule for performance comparison (ref. Equation 14) and also investigate the effects of *sqrt*, *cosine* and *linear* schedules [28, 31, 31] to the performance in Section 5.5. For each method, the experiments are conducted five times and the averaged results are reported. We conduct the student *t-test* for statistical significance test.

5.4 Overall Comparison (RQ1)

The overall performance of our proposed DIFFUREC and the other baselines is presented in Table 2. Here, we can make the following observations.

DIFFUREC consistently outperforms all baselines across the four datasets in terms of all six metrics. In particular, compared with the best baseline, DIFFUREC achieves up to 57.26%/56.72%, 22.76%/34.34%, 10.78%/11.36% and 11.84%/11.39% (HR/NDCG) improvements on *Amazon Beauty*, *Amazon Toys*, *Movielens-1M* and *Steam* datasets, respectively. The significant performance gain

⁸<https://github.com/FeiSun/BERT4Rec>

⁹<https://github.com/THUDM/ComiRec>

¹⁰<https://github.com/THUwangcy/ReChorus/tree/CIKM22>

¹¹https://github.com/noveens/svae_cf

¹²<https://github.com/ACVAE/ACVAE-PyTorch>

¹³<https://github.com/zfan20/STOSA>

Table 2. Experimental results (%) on the four datasets. The best results are in boldface, and the second-best underlined. Symbol ▲% is the relative improvement of DIFFURec against the best baseline. * denotes a significant improvement over the best baseline (t-test $P < .05$).

Dataset	Metric	GRU4Rec	Caser	SASRec	BERT4Rec	ComiRec	TiMiRec	SVAE	ACVAE	STOSA	DifffuRec	▲%
Beauty	HR@5	1.0112	1.6188	3.2688	2.1326	2.0495	1.9044	0.9943	2.4672	<u>3.5457</u>	5.5758*	57.26%
	HR@10	1.9370	2.8166	6.2648	3.7160	4.4545	3.3434	1.9745	3.8832	6.2048	7.9068*	26.21%
	HR@20	3.8531	4.4048	8.9791	5.7922	7.6968	5.1674	3.1552	6.1224	<u>9.5939</u>	11.1098*	15.80%
	NDCG@5	0.6084	0.9758	2.3989	1.3207	1.0503	1.2438	0.6702	1.6858	<u>2.5554</u>	4.0047*	56.72%
	NDCG@10	0.9029	1.3602	<u>3.2305</u>	1.8291	1.8306	1.7044	0.9863	2.1389	3.2085	4.7494*	47.02%
	NDCG@20	1.3804	1.7595	3.6563	2.3541	2.6451	2.1627	1.2867	2.7020	3.7609	5.5566*	47.75%
Toys	HR@5	1.1009	0.9622	4.5333	1.9260	2.3026	1.1631	0.9109	2.1897	4.2236	5.5650*	22.76%
	HR@10	1.8553	1.8317	6.5496	2.9312	4.2901	1.8169	1.3683	3.0749	<u>6.9393</u>	7.4587*	7.48%
	HR@20	3.1827	2.9500	9.2263	4.5889	6.9357	2.7156	1.9239	4.4061	<u>9.5096</u>	9.8417*	3.49%
	NDCG@5	0.6983	0.5707	3.0105	1.1630	1.1571	0.7051	0.5580	1.5604	<u>3.1017</u>	4.1667*	34.34%
	NDCG@10	0.9396	0.8510	3.7533	1.4870	1.7953	0.9123	0.7063	1.8452	<u>3.8806</u>	4.7724*	22.98%
	NDCG@20	1.2724	1.1293	4.3323	1.9038	2.4631	1.1374	0.8446	2.1814	<u>4.3789</u>	5.3684*	22.60%
Movielens-1M	HR@5	5.1139	7.1401	9.3812	13.6393	6.1073	<u>16.2176</u>	1.4869	12.7167	7.0495	17.9659*	10.78%
	HR@10	10.1664	13.3792	16.8941	20.5675	12.0406	<u>23.7142</u>	2.7189	19.9313	14.3941	26.2647*	10.76%
	HR@20	18.6995	22.5507	28.318	29.9479	21.0094	<u>33.2293</u>	5.0326	28.9722	24.9871	36.7870*	10.71%
	NDCG@5	3.0529	4.1550	5.3165	8.8922	3.5214	<u>10.8796</u>	0.9587	8.2287	3.7174	12.1150*	11.36%
	NDCG@10	4.6754	6.1400	7.7277	11.1251	5.4076	<u>13.3059</u>	1.2302	10.5417	6.0771	14.7909*	11.16%
	NDCG@20	6.8228	8.4304	10.5946	13.4763	7.6502	<u>15.7019</u>	1.8251	12.8210	8.7241	17.4386*	11.06%
Steam	HR@5	3.0124	3.6053	4.7428	4.7391	2.2872	<u>6.0155</u>	3.2384	5.5825	4.8546	6.6742*	10.95%
	HR@10	5.4257	6.4940	8.3763	7.9448	5.4358	<u>9.6697</u>	5.8275	9.2783	8.5870	10.7520*	11.19%
	HR@20	9.2319	10.9633	13.6060	12.7332	10.3663	<u>14.8884</u>	7.9753	14.4846	14.1107	16.6507*	11.84%
	NDCG@5	1.8293	2.1586	2.8842	2.9708	1.0965	<u>3.8721</u>	1.8836	3.5429	2.9220	4.2902*	10.80%
	NDCG@10	2.6033	3.0846	4.0489	4.0002	2.1053	<u>5.0446</u>	2.6881	4.7290	4.1191	5.5981*	10.97%
	NDCG@20	3.5572	4.2073	5.3630	5.2027	3.3434	<u>6.3569</u>	3.2323	6.0374	5.5072	7.0810*	11.39%

suggests that DIFFURec can effectively accommodate the four characteristics (*i.e.*, multiple latent aspects of items, multiple interests of users, uncertainty, and guidance of target item) in a unified framework.

Due to the limited capacity of long-term dependency modeling, GRU4Rec and Caser deliver unsatisfied results compared with other baselines across all the datasets. To address this issue, SASRec endeavors to apply uni-directional Transformer to capture more complicated dependency relations for recommendation. Moreover, BERT4Rec believes that future data (*i.e.*, the interacted records after current interaction) is also beneficial for sequential recommendation, thus a bi-directional Transformer is utilized instead. The experiment results illustrate that the Transformer remains an efficient and effective model and achieves better performance than GRU4Rec and Caser in most settings.

On the other hand, the multi-interest models like ComiRec and TiMiRec cannot achieve overwhelming superiority against the conventional sequential neural models across all the datasets, especially compared against SASRec and BERT4Rec. Furthermore, on different size datasets, ComiRec and TimiRec perform differently. We speculate it is because the ComiRec pays more attention to the disparity of sequence representations while ignoring the current intention modeling. In contrast, TiMiRec introduces the target item as auxiliary supervised signals for multi-interest distribution generation when model training. Thus, it achieves much better performance than ComiRec on more complicated and long sequence datasets, *e.g.*, *Movielens-1M* and *Steam*.

As a pioneer work of VAE-based solutions for sequential recommendation, SVAE only utilizes GRU for sequential modeling and reports the worst results. ACVAE introduces adversarial learning via AVB framework to enhance the representation ability and acquire a competitive performance in most settings. Additionally, based on Gaussian distribution and Wasserstein distance modeling, STOSA improves the standard self-attention by exploiting dynamic uncertainty injection and distribution representation. In general, the results of STOSA across all the datasets firmly confirm

the effectiveness of distribution representation learning and uncertainty injection for sequential recommendation.

5.5 Ablation Study (RQ2)

To verify the effectiveness of each design choice in DIFFUREC, we replace one of them each time to analyze the performance variation.

- **w GRU**: replaces the Transformer module with a GRU as the *Approximator* and obtain the representation derived at the last step as $\hat{\mathbf{x}}_0$.
- **w R***: replaces the inner product of rounding operation in Equation 20 with the following variant as in [12]:

$$\frac{2\mathbf{x}_0 \cdot \mathbf{e}_i^T}{||\mathbf{x}_0|| ||\mathbf{e}_i||} - \left(\frac{\mathbf{x}_0}{||\mathbf{x}_0||} + \frac{\mathbf{e}_i}{||\mathbf{e}_i||} \right) \quad (24)$$

- **w Cosine**: replaces the *truncated linear* schedule with *cosine* schedule as in [20] for noise arrangement.
- **w L**: replaces the *truncated linear* schedule with *linear* schedule as in [31] for noise arrangement.
- **w Sqrt**: replaces the *truncated linear* schedule with *sqrt-root* schedule as in [28] for noise arrangement.

Table 3 shows the ablation results on these four datasets. It is encouraging to note that when we replace the *Approximator* from Transformer backbone to GRU for target item representation reconstruction, DIFFUREC still achieves superior performance on *Movielens-1M* and *Steam* datasets while the space and time complexity drop significantly. And this phenomenon is also observed in CV domain that compared with other sophisticated model architectures, a U-Net (a simple Encode-Decoder framework with convolution operation) combined with Diffusion could also achieve extraordinary results. Additionally, we also select the rounding strategy proposed in [12] as an alternative for target item prediction, but *HR/NDCG* reduce by at least 68.71%/74.61%, 92.40%/95.64%, 75.08%/75.93% and 74.12%/76.17% on *Amazon Beauty*, *Amazon Toys*, *Movielens-1M* and *Steam* datasets, respectively. We believe it is because inner product operation, as most works leveraged, is aligned with objective loss and may be more suitable for sequential recommendation [4, 25, 42].

Apart from that, we investigate the impact of different schedules to the final results. Following [31], we plot the value of $\bar{\alpha}_t$ under different schedules as shown in Figure 4, where $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$, $\alpha_s = 1 - \beta_s$ (ref. Equation 3). We could find that the Truncated Linear schedule provides a near-linear sharp drop in the middle of the process and subtle changes near the extremes of $t = T$ compared with other schedules. This design proves to be effective in enhancing the model's capability of interest modeling since we aim to recover the original input from noise via only one-step estimation (ref. Algorithm 1). Consequently, a sharp corruption (*i.e.*, add more noise) in the training process is necessary for better denoising ability. However, we also find that in some cases the Linear schedule shows superiority over others, indicating some better alternatives that fall between Truncated Linear and Linear to be explored in the future. Besides, as [17] emphasizes that different schedules will impact the final outcomes to some extent but could not acquire huge margin fluctuations. Our experiments also manifest this conclusion (see Figure 3) that the truncated linear schedule achieves the best results against other schedules on *Amazon Beauty* and *Amazon Baby* datasets but attain sub-optimal performance on the other two datasets. Moreover, following [20, 28], we explore the root mean-squared error loss (RMSE) and ϵ prediction via *Approximator*

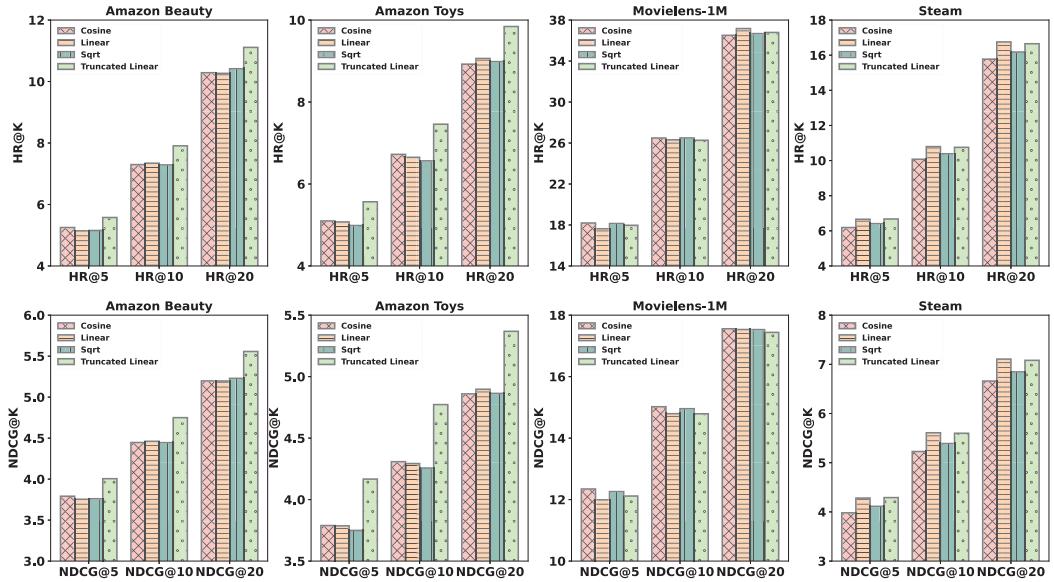


Fig. 3. Performance of DIFFURec with different schedule.

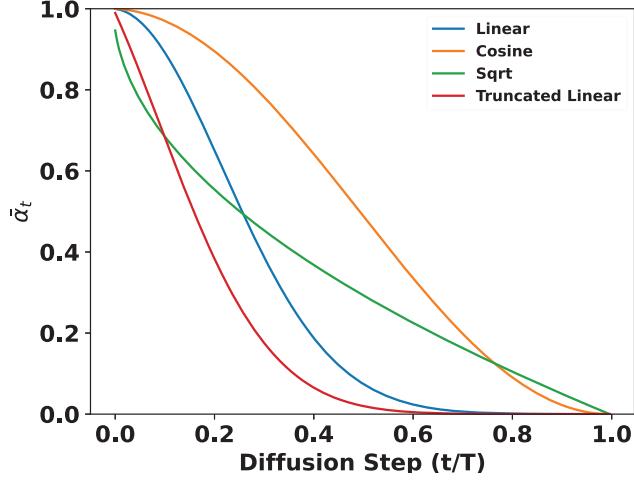
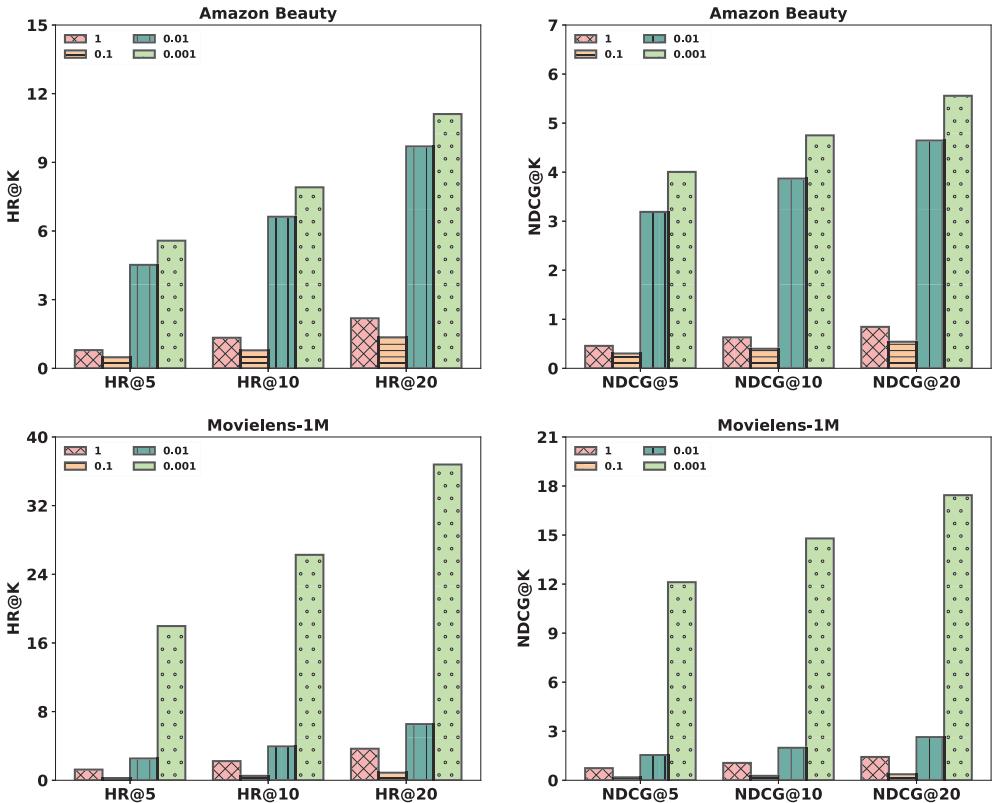
Table 3. Results (%) of ablation experiments. The best results are in boldface, and the second-best underlined.

Dataset	Ablation	HR@5	HR@10	HR@20	NDCG@5	NDCG@10	NDCG@20
Beauty	w GRU	<u>3.1773</u>	<u>4.6685</u>	<u>6.9000</u>	<u>2.2253</u>	<u>2.7075</u>	<u>3.2682</u>
	w R*	1.3036	2.2902	3.4768	0.7964	1.1130	1.4108
	DIFFURec	5.5758	7.9068	11.1098	4.0047	4.7494	5.5566
Toys	w GRU	<u>2.3895</u>	<u>3.4150</u>	<u>4.9407</u>	<u>1.7577</u>	<u>2.0849</u>	<u>2.4700</u>
	w R*	0.1194	0.4133	0.7480	0.0567	0.1510	0.2343
	DIFFURec	5.5650	7.4587	9.8417	4.1667	4.7724	5.3684
Movie	w GRU	<u>16.6016</u>	<u>24.6094</u>	<u>34.9772</u>	<u>10.9553</u>	<u>13.5348</u>	<u>16.1403</u>
	w R*	4.0690	6.0872	9.1688	2.7787	3.4297	4.1978
	DIFFURec	17.9659	26.2647	36.7870	12.1150	14.7909	17.4386
Steam	w GRU	<u>6.2832</u>	<u>10.1723</u>	<u>15.8169</u>	<u>4.0056</u>	<u>5.2548</u>	<u>6.6727</u>
	w R*	1.4873	2.5977	4.3086	0.9022	1.2582	1.6877
	DIFFURec	6.6742	10.7520	16.6507	4.2902	5.5981	7.0810

for target item representation reconstruction (*i.e.*, instead of predicted the target item representation \hat{x} straightforwardly. Specifically, we count on predicted ϵ for target item representation reconstruction, $\hat{x}_0 = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon)$), but the training loss can not converge. Therefore, we do not report the final results here.

5.6 Impact of Hyper-parameter Setting (RQ3)

Besides the noise schedule, the value of λ in Equation 13 also regulates the discrimination capacity of the item representations. Here, we set different δ which controls the mean and variance of λ sampling distribution and investigate the effect of this hyper-parameter on the performance of DIFFURec. Figure 5 presents the performance patterns by varying λ values. We can see that a small λ generally resulted in better performance on both *Amazon Beauty* and *MovieLens-1M* datasets, while when we increase the λ to 0.1, the *HR* and *NDCG* drop sharply, especially on *MovieLens-1M*.

Fig. 4. $\bar{\alpha}_t$ throughout diffusion in different schedule.Fig. 5. Parameter sensitivity of λ .

dataset. We suppose that a large λ may add too much noise to the historical interaction sequences,

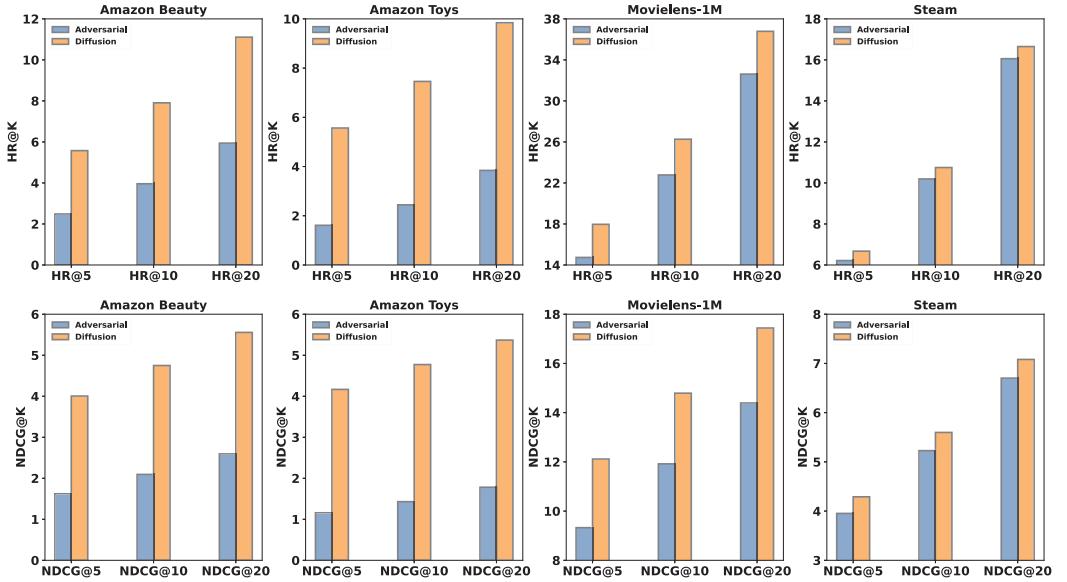


Fig. 6. Performance of adversarial training and DIFFUREC.

which will corrupt the original information and hinder the model to precisely understanding the user's interest.

5.7 Compared with Adversarial Training (RQ4)

To further analyze the effectiveness of uncertainty and diversity of diffusion model to the recommendation, we also adopt adversarial training for performance comparison. As described in Section 4.6, by adding adversarial perturbations on the training process, the model's robustness and performance can be improved. We set $\epsilon = 0.5$ and $\gamma = 1$. The results are shown in Figure 6.

Observing Figure 6 we could find that the adversarial Transformer achieves better results against BERT4Rec, albeit the model architectures are the same. Specifically, adversarial Transformer improves the baseline on *Amazon Beauty* and *MovieLens-1M* datasets w.r.t. *HR* and *NDCG* by at least 2.57%/10.23% and 8.05%/4.89%, respectively. It proves that the perturbation on the adversarial training process will be helpful for model robustness and performance improvement. However, compared with DIFFUREC, our method is superior to the adversarial Transformer by a large margin on all the datasets. One explanation is that the injected uncertainty and perturbation in diffusion and reverse phases is not just a pure noise randomly sampled from a Gaussian distribution for model robustness argumentation, but some auxiliary information from target item representation will also be introduced as supervised signals to facilitate model to capture user's current intentions for final prediction, which may be the recipe of DIFFUREC could achieve best results.

5.8 Performance on Varying Lengths of Sequences and Items with Different Popularity (RQ5)

We further analyze the performance of DIFFUREC against other competitive baselines (SASRec, TimiRec and STOSA) on different lengths of sequences as well as on head and long-tail items. Figure 7 shows the distribution of sequence length and item interaction times on *Amazon Beauty* and *MovieLens-1M* datasets, respectively. We could find that for *Amazon Beauty* dataset, the length

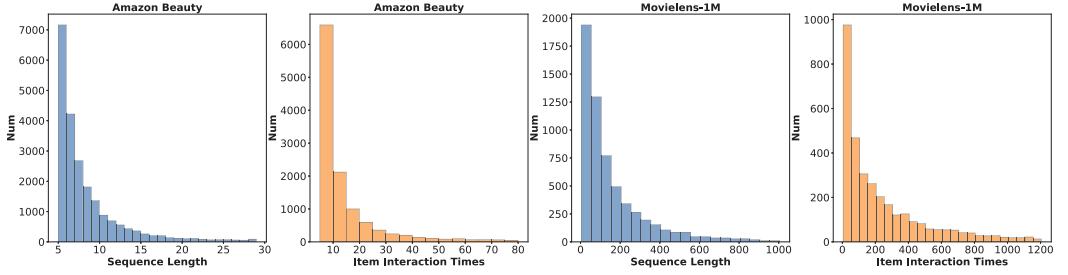


Fig. 7. Frequency histogram of sequence length and item interaction times for *Amazon Beauty* and *MovieLens-1M* dataset. The maximum length and maximum interactions are 204/431 and 2341/3428 respectively for *Amazon Beauty* and *MovieLens-1M* datasets.

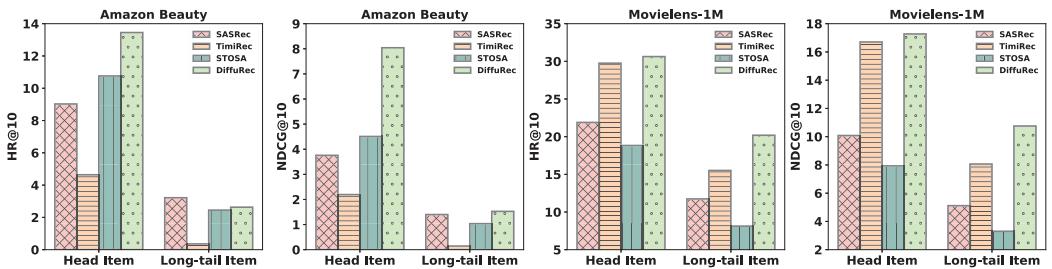


Fig. 8. Performance on the head and long-tail items.

is less than 15 for most of the sequences and the interaction times are less than 40 for most of the items. So as to the *MovieLens-1M* dataset, although the sequence length and interaction numbers far exceed *Amazon Beauty* dataset, the short sequence and long-tail items are still the majority.

Head and Long-tail Items. Following previous works [57], we split the first 20% most frequent items as head items and denote the rest are long-tail items, guided by Pareto Principle [2], for performance evaluation. The results are shown in Figure 8. We could find that for both *Amazon Beauty* and *MovieLens-1M* datasets, the performance on head items is significantly superior to the long-tail items for all the models. However, this gap is notably smaller on the *MovieLens-1M* dataset, where the performance difference between head items and long-tail items is less than three times that of *Amazon Beauty* dataset, where it is more than five times greater. Given that the number of items on *MovieLens-1M* is rather small, this might be the reason why there is no obvious performance difference between head items and long-tail items. Overall, DIFFURec outperforms all baselines on most of the settings, in particular in *NDCG* metric.

Different Sequence Length. Based on the percentile of sequence length l , we split the *Amazon Beauty* / *MovieLens-1M* datasets into five groups: short ($0 < l \leq 5$ / $0 < l \leq 37$), mid-short ($5 < l \leq 6$ / $37 < l \leq 70$), medium ($6 < l \leq 7$ / $70 < l \leq 126$), mid-long ($7 < l \leq 10$ / $126 < l \leq 253$), and long ($l > 10$ / $l > 253$). The performance of different baselines on varied sequence lengths is depicted in Figure 9, it can be observed that the results on *MovieLens-1M* dataset and *Amazon Beauty* dataset are completely opposite. As for *Amazon Beauty* dataset, DIFFURec achieves a better performance on long sequences. Conversely, for the *MovieLens-1M* dataset, as the sequence length increases, there will be a gradual decline in performance. Comparing the sequence length of these two datasets, we could find that 80% of sequences are shorter than 10 for *Amazon Beauty* dataset whereas for the

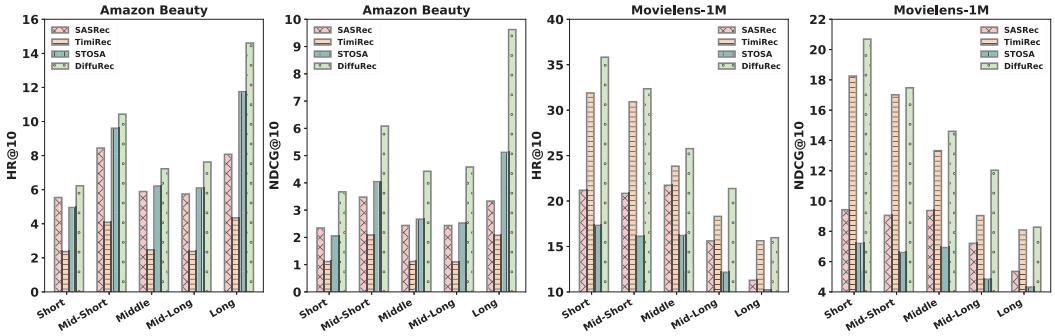
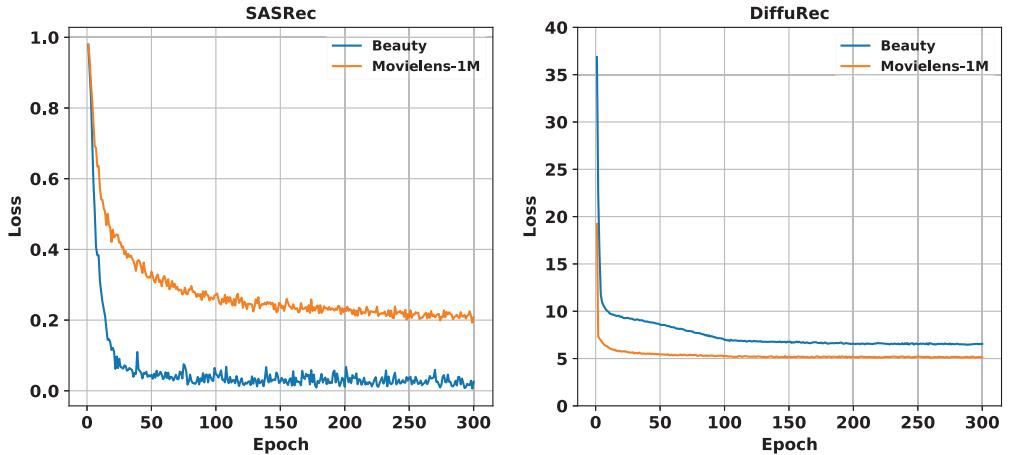


Fig. 9. Performance on different length of sequence.

Fig. 10. Curve of training loss on *Amazon Beauty* and *Movielens-1M* datasets

Movielens-1M dataset 80% of sequences are longer than 37. We believe the short sequences can not provide sufficient information to reveal the user's current preference for a precise recommendation, whereas it is also challenging for all the models to handle very long sequences and recommendation. In general, compared with different baselines, our model achieves the best results in all the settings, which manifests that DIFFUREC is robustness to the sequence length.

5.9 Convergence and Efficiency (RQ6)

DIFFUREC is a new paradigm for sequential recommendation which is also an iterative process in reverse and model inference. We thereby analyze its convergence on training process and efficiency for inference. All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU and Intel Xeon CPU E5-2680 v3.

Convergence. We compare the convergence speed of our model with SASRec, a representative sequential recommendation baseline, whose model structures are also similar to ours (*i.e.*, Transformer-based). For a fair comparison, we keep the optimizer, dropout ratio, hidden size, multi-head numbers, and Transformer block numbers as the same, *i.e.*, Adam, 0.1, 128, 4, and 4 respectively.

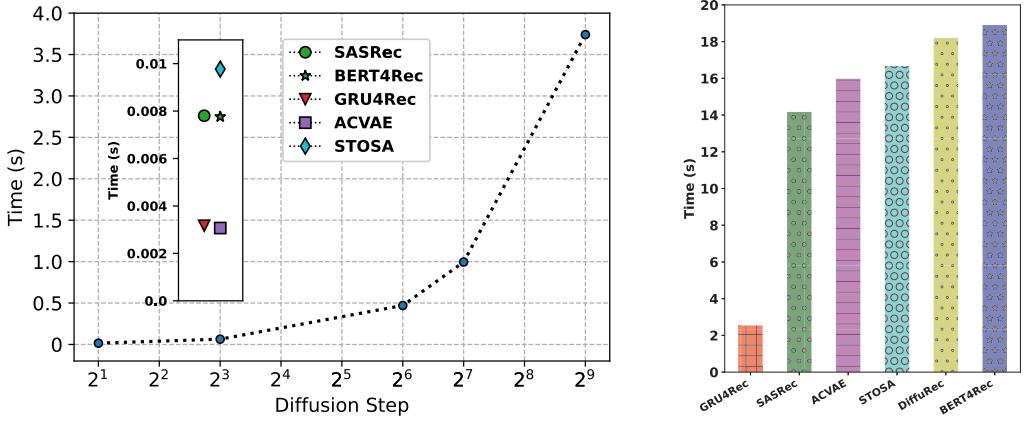


Fig. 11. Inference time (in seconds for one sample) on *Amazon Beauty* dataset (left). The points in the inset box present the inference time of baselines. The folding line chart reveals the inference time for the different reverse steps. Average training time (in seconds for one batch) on *Amazon Beauty* dataset (right).

Table 4. Time complexity of DIFFUREC and other baselines. d and n are the representation dimension and sequence length respectively.

Model	GRU4Rec	SASRec	BERT4Rec	ACVAE	STOSA	DiffuRec
Complexity	$O(nd^2)$	$O(nd^2 + dn^2)$	$O(nd^2 + dn^2)$	$O(nd^2)$	$O(nd^2 + dn^2)$	$O(nd^2 + dn^2)$

Figure 10 depicts the loss curves on training process. Observe that on *Amazon Beauty* dataset both DIFFUREC and SASRec could converge after 150 training epochs. But on *Movielens-1M* dataset, DIFFUREC presents faster convergence (around 100 epochs) than SASRec (around 250 epochs). We speculate it is because the average length of *Movielens-1M* dataset (95.19) is over ten times than *Amazon Beuaty* dataset (8.53), SASRec thereby could not capture the long-term dependency between items efficiently, while our model can induce the user's current interest extracted from the target item representation as auxiliary information, which accelerates the training convergence speed of models.

Efficiency. In general, the number of reverse steps is the most critical factor to determine the inference speed of diffusion models. Hence, we investigate the effect of reverse step to inference time and also analyze the time complexity with big O notation of DIFFUREC against other recipes. The results are shown in Figure 11 and Table 4.

Observing Figure 11 (left), the inference time of DIFFUREC is close to SASRec and BERT4Rec when we set the total diffusion step as a small value (*i.e.*, 2), as all of them are Transformer-based models and the time complexity is the same (ref. Tabel 4). Besides, with the increase of reverse step, the inference time for one sample rises exponentially, while the performance also exhibits a fluctuation (observing Figure 12). Balancing the quality of representation and the time, we consider a moderate reverse step (*e.g.*, 32 or 64) is sufficient for SR. Furthermore, compared with Transformer-based methods, both GRU4Rec and ACVAE apply GRU as the backbone for recommendation. Hence, their inference time and time complexity are similar and shorter than Transformer-based solutions. STOSA improves Transformer as a stochastic model and also adopts the Wasserstein distance as an

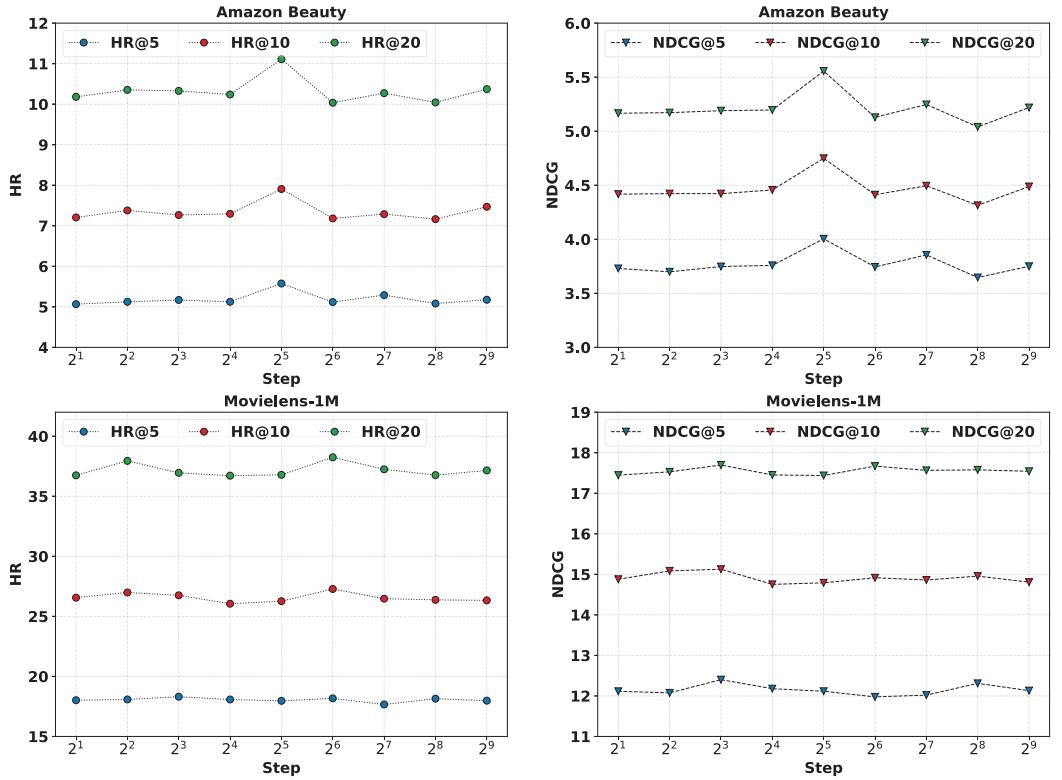


Fig. 12. Recall@K and NDCG@K at different reverse steps on *Amazon Beauty* and *Movielens-1M* datasets.

alternative to the inner product for discrepancy measure between two items. Consequently, the inference time is slightly longer than the unmodified Transformer-based methods.

We further compare the average training time of one epoch between DIFFUREC and other baselines, as shown in Figure 11 (right). We could find that the training time of GRU4Rec is significantly less than others since the model structure and size are more lightweight than the Transformer-based architectures. Noting that the training time of SASRec, ACVAE, STOSA, DIFFUREC, and BERT4Rec are very close, with all models encompassing a range from 14s to 18s. Thus, we believe the training process of our DIFFUREC will not be the bottleneck of model optimization.

5.10 Visualization of Uncertainty and Diversity (RQ7)

Uncertainty. As we emphasize that attributed to the theoretical underpinnings of Diffusion framework, DIFFUREC could inject some uncertainty in diffusion and reverse process, which will be beneficial to the final performance improvement. We then analyze the uncertainty of the reconstructed target item representation \mathbf{x}_0 reversed by our method. Figure 13 plots a set of reversed \mathbf{x}_0 vectors and other item representation from *Amazon Beauty* and *Movielens-1M* datasets projected into a two-dimensional space via t-SNE [46].

Observing Figure 13, we find that the reversed representation \mathbf{x}_0 from random sequences are dispersed throughout the whole space uniformly. In contrast, the reversed \mathbf{x}_0 of the exemplar sequence under different Gaussian noises are relatively much closer but also hold certain deviations to each other. Actually, most existing solutions are deterministic for a specific sequence. The merit

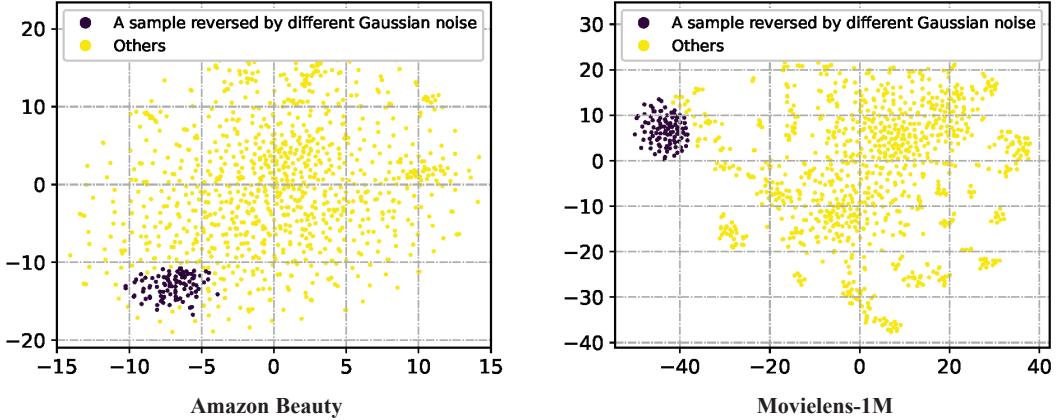


Fig. 13. A t-SNE plot of the reconstructed x_0 representation from reverse phase. The purple points represent the same item but reversed by 100 different Gaussian noise for a specific historical sequence. The yellow points are the others each of which is reversed from a random sequence in *Amazon Beauty* and *Movielens-1M* datasets respectively.

Table 5. The category of recommended Top 5 items. Items sharing the same color indicate they are in similar categories.

Model	Categories of Top 5 Items.	Target Item Category
SASRec	Kids' Electronics, Electronic Toys, <i>Learning & Education</i> , Toy Remote Control & Play Vehicles, Play Vehicles	Electronic Toys
TimiRec	Toy Remote Control & Play Vehicles, Electronic Toys, Play Vehicles, Remote & App Controlled Vehicle Parts, Play Trains & Railway Sets	Electronic Toys
STOSA	<i>Learning & Education</i> , Playsets & Vehicles, <i>Board Games</i> , Train Sets, <i>Wind-up Toys</i>	Electronic Toys
DIFFUREC	Electronic Toys, Helicopter Kits, Remote & App Controlled Vehicles, <i>Card Games</i> , <i>Electronic Software & Books</i>	Electronic Toys

of this uncertainty is much desired for the retrieval stage. Specifically, by reversing 100 different x_0 for the exemplar sequence, the number of unique items that are ranked in their top-20 reaches 643 and 82 respectively for *Amazon Beauty* and *Movielens-1M* datasets. That is, we can generate different reversed x_0 with parallel computing, consequently leading to the diversified retrieval results.

Diversity. To verify the impact of uncertainty on the diversity of final recommendation results, we randomly sample one sequence from *Amazon Toys* dataset and collect the categories of the recommended items. Table 5 depicts the top five items' categories yielded by our DIFFUREC and other alternatives.

As shown in Table 5, SASRec and TimiRec are inclined to recommend items from similar categories, e.g., Electronic Toys, and Vehicles, given that TimiRec leverages the target item as the supervised signal for recommendation, which will constrain the final outcomes to the similar categories inherently. On the contrary, attributed to the uncertainty injection, STOSA, and DIFFUREC could recommend a wider range of categories rather than focusing on the "Electronic Toys". As such, we believe injecting some uncertainty in recommendation will facilitate more flexibility and therefore yield more diversity and richer recommendation results.

6 CONCLUSION

This work focuses on sequential recommendation. We consider that the single vector based representation used in current solutions does not well capture the dynamics in the recommendation context. To model items' latent aspects and users' multi-level interests, we consider Diffusion model to be a good fit. To instantiate this idea, we adapt the Diffusion model to sequential recommendation for the very first time and propose DIFFUREC, which is our key contribution. Specifically, we detail how the diffusion and reverse phase are carried out under the new problem setting, and how to design an Approximator. Experimental results demonstrate the superiority of DIFFUREC on four real-world datasets. We have also conducted extensive experiments to verify the effectiveness of the designed components in DIFFUREC. As a new paradigm to recommendation tasks, different ways and other recommendation scenarios (e.g., session-based recommendation, click-through rate prediction) of adapting the powerful Diffusion model remain under-explored. We hope this work may shed light along this direction.

REFERENCES

- [1] Omri Avrahami, Dani Lischinski, and Ohad Fried. 2022. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18208–18218.
- [2] George EP Box and R Daniel Meyer. 1986. An analysis for unreplicated fractional factorials. *Technometrics* 28, 1 (1986), 11–18.
- [3] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. 2020. Learning gradient fields for shape generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III* 16. Springer, 364–381.
- [4] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2942–2951.
- [5] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. 2020. WaveGrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713* (2020).
- [6] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.
- [7] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. 2022. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089* (2022).
- [8] Yujuan Ding, Yunshan Ma, Wai Keung Wong, and Tat-Seng Chua. 2021. Leveraging two types of global graph for sequential fashion recommendation. In *Proceedings of the 2021 International Conference on Multimedia Retrieval*. 73–81.
- [9] Ziwei Fan, Zhiwei Liu, Shen Wang, Lei Zheng, and Philip S Yu. 2021. Modeling sequences as distributions with uncertainty for sequential recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3019–3023.
- [10] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S Yu. 2022. Sequential recommendation via stochastic self-attention. In *Proceedings of the ACM Web Conference 2022*. 2036–2047.
- [11] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S Yu. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 433–442.
- [12] Shanshan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933* (2022).
- [13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
- [14] Lei Guo, Li Tang, Tong Chen, Lei Zhu, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2021. DA-GCN: a domain-aware attentive graph convolution network for shared-account cross-domain sequential recommendation. *arXiv preprint arXiv:2105.03300* (2021).
- [15] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2022. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432* (2022).
- [16] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *The 41st International ACM SIGIR conference on research & development in information retrieval*. 355–364.

- [17] Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2022. DiffusionBERT: Improving Generative Masked Language Models with Diffusion Models. *arXiv preprint arXiv:2211.15029* (2022).
- [18] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.
- [19] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
- [20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.
- [21] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. 2022. Cascaded Diffusion Models for High Fidelity Image Generation. *J. Mach. Learn. Res.* 23 (2022), 47–1.
- [22] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. 2022. Video diffusion models. *arXiv preprint arXiv:2204.03458* (2022).
- [23] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. *arXiv preprint arXiv:2102.05379* 3, 4 (2021), 5.
- [24] Neil Hurley and Mi Zhang. 2011. Novelty and diversity in top-n recommendation—analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)* 10, 4 (2011), 1–30.
- [25] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
- [26] Walid Krichene and Steffen Rendle. 2022. On sampled metrics for item recommendation. *Commun. ACM* 65, 7 (2022), 75–83.
- [27] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 2615–2623.
- [28] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. 2022. Diffusion-LM Improves Controllable Text Generation. *arXiv preprint arXiv:2205.14217* (2022).
- [29] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.
- [30] Rabeeh Karimi Mahabadi, Jaesung Tae, Hamish Ivison, James Henderson, Iz Beltagy, Matthew E Peters, and Arman Cohan. 2023. TESS: Text-to-Text Self-Conditioned Simplex Diffusion. *arXiv preprint arXiv:2305.08379* (2023).
- [31] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*. PMLR, 8162–8171.
- [32] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* (2022).
- [33] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.
- [34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.
- [35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III* 18. Springer, 234–241.
- [36] 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.
- [37] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. 2022. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).
- [38] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).
- [39] Weiqi Shao, Xu Chen, Long Xia, Jiashu Zhao, and Dawei Yin. 2022. Sequential Recommendation with User Evolving Preference Decomposition. *arXiv preprint arXiv:2203.16942* (2022).
- [40] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*. PMLR, 2256–2265.
- [41] Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, et al. 2022. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236* (2022).

- [42] 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.
- [43] Qiaoyu Tan, Jianwei Zhang, Ninghao Liu, Xiao Huang, Hongxia Yang, Jingren Zhou, and Xia Hu. 2021. Dynamic memory based attention network for sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4384–4392.
- [44] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.
- [45] Yu Tian, Jianxin Chang, Yanan Niu, Yang Song, and Chenliang Li. 2022. When Multi-Level Meets Multi-Interest: A Multi-Grained Neural Model for Sequential Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1632–1641.
- [46] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
- [47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [48] Chenyang Wang, Zhefan Wang, Yankai Liu, Yang Ge, Weizhi Ma, Min Zhang, Yiqun Liu, Junlan Feng, Chao Deng, and Shaoping Ma. 2022. Target Interest Distillation for Multi-Interest Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2007–2016.
- [49] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, ShaoZhang Niu, and Jimmy Huang. 2020. KERL: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 209–218.
- [50] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 328–337.
- [51] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.
- [52] Teng Xiao, Shangsong Liang, and Zaiqiao Meng. 2019. Hierarchical neural variational model for personalized sequential recommendation. In *The World Wide Web Conference*. 3377–3383.
- [53] Zhe Xie, Chengxuan Liu, Yichi Zhang, Hongtao Lu, Dong Wang, and Yue Ding. 2021. Adversarial and contrastive variational autoencoder for sequential recommendation. In *Proceedings of the Web Conference 2021*. 449–459.
- [54] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2022. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796* (2022).
- [55] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. 2020. Future data helps training: Modeling future contexts for session-based recommendation. In *Proceedings of The Web Conference 2020*. 303–313.
- [56] Shengyu Zhang, Lingxiao Yang, Dong Yao, Yujie Lu, Fuli Feng, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2022. Re4: Learning to Re-contrast, Re-attend, Re-construct for Multi-interest Recommendation. In *Proceedings of the ACM Web Conference 2022*. 2216–2226.
- [57] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. 2021. A model of two tales: Dual transfer learning framework for improved long-tail item recommendation. In *Proceedings of the web conference 2021*. 2220–2231.
- [58] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. 2012. Auralist: introducing serendipity into music recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 13–22.
- [59] Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 33. 5885–5892.
- [60] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.
- [61] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is all you need for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*. 2388–2399.
- [62] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2013. Using temporal data for making recommendations. *arXiv preprint arXiv:1301.2320* (2013).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009