DiffuRec: A Diffusion Model for Sequential Recommendation

Zihao Li
Wuhan University
Wuhan, China
zihao.li@whu.edu.cn

Aixin Sun
Nanyang Technological University
Singapore
axsun@ntu.edu.sg

Chenliang Li*
Wuhan University
Wuhan, China
cllee@whu.edu.cn

ABSTRACT

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. Afterwards, the item representation is fed into an Approximator for target item representation reconstruction. In reversion phase, based on user's historical interaction behaviors, we reverse a Gaussian noise into the target item representation, then apply rounding operation for target item prediction. Experiments over four datasets show that DiffuRec outperforms strong baselines by a large margin.

CCS CONCEPTS

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

KEYWORDS

Diffusion Model, Sequential Recommendation, User Preference Learning

1 INTRODUCTION

Sequential Recommendation (SR) aims to modeling 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 [13, 14]. Recent efforts illustrated the remarkable ability of self-attention mechanism for long-term dependency capturing. Thus, self-attention based methods became a prominent solution [20, 32, 40, 43]. SASRec [20] is a pioneering work that applies uni-direction Transformer to learn transition patterns of items in sequences. BERT4Rec [32] adopted bi-direction Transformer encoder to predict next item. Based on information propagation and aggregation, Graph Neural Network (GNN) has also achieved impressive results on SR [8, 11]. Compared to earlier solutions, GNN could capture high-order dependencies between items.

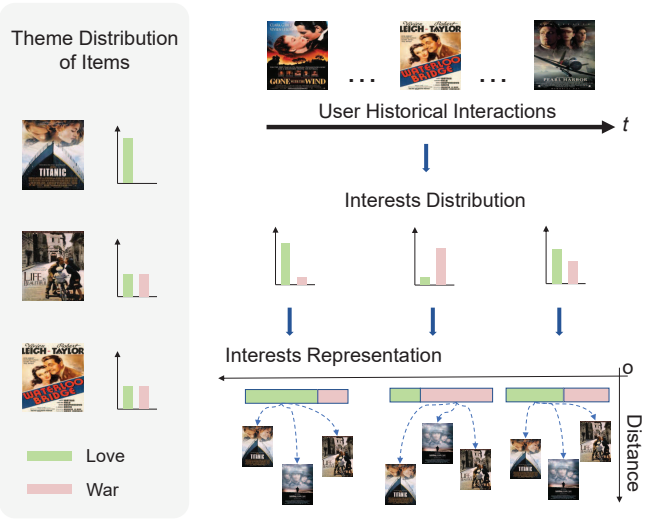

Figure 1: An example of multiple interests of users and multiple aspects of items.

All these mainstream methods learn item representation as an embedding vector. 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.

Multiple Interests. User's interests and preferences are diverse and inconstant. Further, user may dynamically adjust her interests based on different scenarios or periods (circled by the right box). 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.

*Chenliang Li is the corresponding author.

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 [6, 19, 45]. 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 [38]. 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 for ranking stage.

To partially address the aforementioned characteristics, attention mechanism and dynamic routing for multi-interest modeling and recommendation have been explored [2, 22, 38, 47]. 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 [42] and collapse issue of posterior distribution approximation [24, 46], 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 many fields [23, 26]. 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 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. 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 [23], 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 latent aspects and user's diverse intentions as distributions.
- We fuse the target item embedding for historical interacted item representation generation. Thus, user's current preference could be introduced as an auxiliary signal for target item prediction, and some uncertainty could also be injected.
- 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.

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 brief the development of diffusion models.

Sequential Recommendation. Markov Chain played a crucial role in sequential recommendation in the early stage [27, 30, 49], for modeling a user's interaction behavior as a Markov Decision Process (MDP). Later, deep sequential neural network, *e.g.,* GRU, LSTM, and further variants [14, 41], delivered promising performance on this task. Afterwards, other advancements of sequence modeling, *e.g.,* attention mechanisms [20, 32, 34] and memory network [4, 33], demonstrated their effectiveness with state-of-the-art performance in this area. Attributed to the information propagation and aggregation, graph neural networks are more effective to incorporate high-order relationship in a sequence, and were also utilized for sequential recommendation [5, 8, 11].

Multi-Interest Modeling. As user's interests are dynamic and diverse, multi-interest modeling with soft-attention mechanism and dynamic routing became prominent for sequential recommendation. For example, DIN [47], MIND [22], and ComiRec [2] are a few representative efforts in this line. On top of that, some other work, *e.g.,* Re4 [44], TiMiRec [38] proposed auxiliary loss functions for target interest distillation and distinction. MGNM [35] combined graph convolution operation with dynamic routing for multi-interest learning. However, all these solutions require to predefine 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 to sub-optimal performance.

Representation Distribution and Uncertainty Modeling. VAE, with its ability to model probabilistic latent variables, has been used for generating distributions and injecting uncertainty. For example, Liang *et al.* proposed Multi-VAE [24], 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 [29]. Considering the posterior distribution approximation hurts the representation ability of conventional VAE methods, ACVAE introduced the adversarial variational bayes (AVB) framework to enhance the generation of latent variables [42]. Wang *et al.* further

incorporated contrastive learning into the VAE paradigm to alleviate the representation degeneration problem [39]. Different from these VAE methods, Fan *et al.* [6, 7] proposed a novel stochastic self-attention (STOSA) module very recently. They directly represent each item's embedding as a stochastic Gaussian distribution for uncertainty modeling and sequential recommendation. Nevertheless, VAE methods struggle with representation degeneration and collapse issues despite their ability to model latent variables as distributions.

Diffusion Models. Motivated by non-equilibrium thermodynamics [31], Diffusion model have shown its great potential in computer vision [1, 16, 17], natural language processing [9, 12, 23] and other areas [3, 26]. Diffusion-LM [23] 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 [9]. In spite of the ubiquity of Diffusion models in other domains, adapting the model to recommender systems remains underexplored. We consider Diffusion model is a good fit to SR, and make the first attempt in this paper.

3 PRELIMINARY

Before presenting our model, we briefly introduce the diffusion model as preliminary knowledge. Endowed with the property of latent variable modeling, diffusion model is capable of continuous representation generation by using a diffusion and reverse phase.

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.,* $\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow, \ldots, \rightarrow \mathbf{x}_t$) as follows:

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

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

The value of β_t is generated from a pre-defined noise schedule β controlling how much noise is injected at which step. The common noise schedule includes square-root [23], cosine [15], and linear [25]. According to [15], at an arbitrary diffusion step t, we can directly derive $\mathbf{x}_t$ conditioned on the input $\mathbf{x}_0$. Then Equation 1 can be rewritten as follows:

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

$$\overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s, \ \alpha_s = 1 - \beta_s. \tag{3}$$

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

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

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

$$\tilde{\beta}_s = \frac{1-\overline{\alpha}_{s-1}}{1-\overline{\alpha}_s}\beta_s \tag{6}$$

As $\mathbf{x}_0$ is unknown in the reverse phase, a deep neural network (*e.g.,* Transformer [37] or U-Net [28]) is generally used for estimating $\mathbf{x}_0$. Given the original representation $\mathbf{x}_0$ and the schedule β, we optimize the reverse phase, *i.e.,* $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, to minimize the following variational lower bound (VLB):

$$\begin{aligned}\mathcal{L}_{vlb} = & \underbrace{\mathbb{E}_q\left[D_{KL}\left(q(\mathbf{x}_t|\mathbf{x}_0)||p(\mathbf{x}_t)\right)\right]}_{L_t} \\ & + \underbrace{\mathbb{E}_q\left[\sum_{s=2}^{t} D_{KL}(q(\mathbf{x}_{s-1}|\mathbf{x}_s, \mathbf{x}_0)||p_\theta(\mathbf{x}_{s-1}|\mathbf{x}_s))\right]}_{L_{t-1}} \\ & \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\end{aligned} \tag{7}$$

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 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 reversion 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. Following [15], we can resort to the mean-squared error loss as follows:

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

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.

4 METHODOLOGY

We first formally define the task of sequential recommendation, then we detail the proposed DiffuRec model.

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 $\mathcal{S} = [i_1^u, i_2^u, ..., i_n^u]$. 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 ranking list of items as the candidates for next item that will be preferred by the user.

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 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

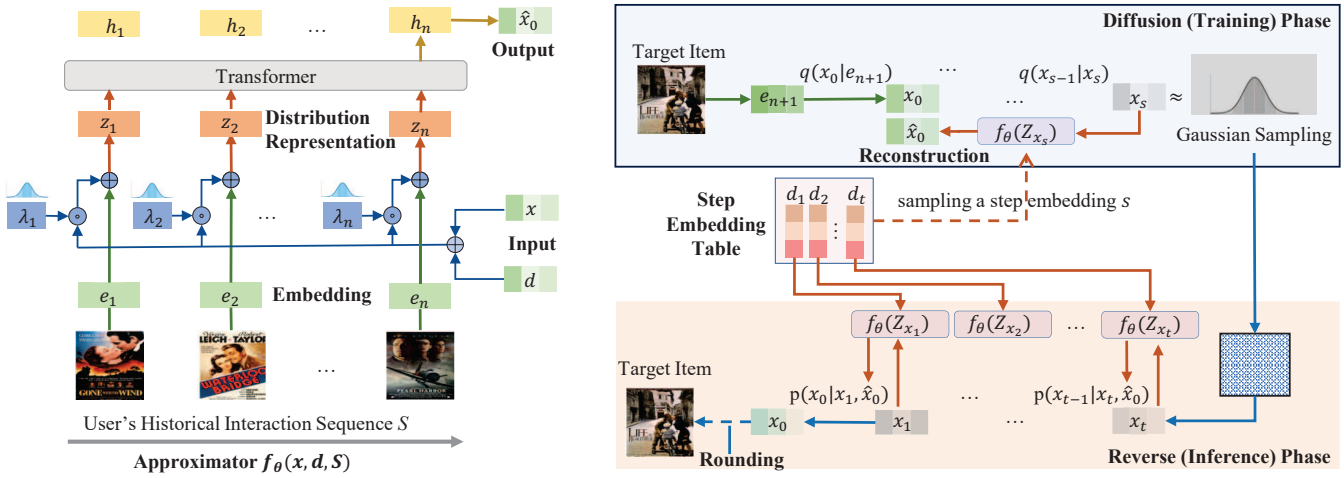

Figure 2: Architecture of DiffuRec. The figure on the left is the *Approximator*, a Transformer backbone for target item representation reconstruction. The two figures on the right illustrate the diffusion phase and the reverse phase, respectively.

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 [10], the injected noise can help us to derive sharper item embeddings, such that $\mathbf{x}_s$ 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 $\mathbf{Z}_{x_s} = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n]$ 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 $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Similar to the diffusion process, the resultant representations $\mathbf{Z}_{x_t}$ adjusted by using $\mathbf{x}_t$ are fed into the well-trained approximator f_θ 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 $\mathbf{x}_0$ for target item $\hat{i}_{n+1}$ prediction.

Formally, the above process can be formulated as follows:

Diffusion:

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

Reverse:

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

Rounding:

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

4.2 Approximator

Following [9, 23], we use Transformer as Approximator $f_\theta(\cdot)$ to generate $\hat{\mathbf{x}}_0$ in diffusion and reversion phases.

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

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

where symbol $\odot$ is the element-wise product, and $\mathbf{d}$ is the embedding of the corresponding diffusion/reverse step created by following [37]. Through step embeddings, the specific information of each diffusion and reverse step could be captured. As discussed earlier, in diffusion phase, $\mathbf{x}$ is the noised target item embedding $\mathbf{x}_s$ (more discussion in Section 4.3). In contrast, $\mathbf{x}$ is the reversed target item representation, *i.e.,* $\mathbf{x} = \mathbf{x}_s$ $(s = t, t-1, \ldots, 2, 1)$. In Equation 13, $\lambda_\mathbf{i}$ is sampled from a Gaussian distribution, *i.e.,* $\lambda_i \sim \mathcal{N}(\delta, \delta)$, δ is a hyperparameter. 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 [37], *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. Following [18, 31], a uniform noise schedule was explored for the discrete domain. Hence, we utilize a truncated linear schedule for β_s generation as follows:

$$\beta_s = \frac{a}{t}s + \frac{b}{s} \tag{14}$$

Algorithm 1: Reverse or Inference

Input:
Historical Sequence: $(i_1, \cdots, i_n)$;
Target item Representation $\mathbf{x}_t$: $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
Total Reverse Steps: t;
Schedule β: *Linear*;
Approximator: $f_\theta(\mathbf{x})$;
Output:
Predicted Target Item: i_{n+1};
while $t > 0$ **do**
$(\mathbf{z}_1, ..., \mathbf{z}_n) \leftarrow (\mathbf{e}_1 + \lambda_1(\mathbf{x}_t + \mathbf{d}_t), ..., \mathbf{e}_n + \lambda_n(\mathbf{x}_t + \mathbf{d}_t)), ;$ `// Distribution Representation`
$\hat{\mathbf{x}}_0 \leftarrow f_\theta(\mathbf{z}_1, ..., \mathbf{z}_n)$; `// Reconstruction`
$\mathbf{x}_{t-1} \leftarrow \tilde{\mu}(\hat{\mathbf{x}}_0, \mathbf{x}_t) + \tilde{\beta}_t * \epsilon'$, $(\epsilon' \sim \mathcal{N}(\mathbf{0}, \mathbf{I}))$; `// Reversion`
$t = t - 1$; `// Iteration`
end while
$i_{n+1} \leftarrow \text{Rounding}(\mathbf{x}_0)$;

Algorithm 2: Diffusion or Training

Input:
Historical Sequence: $(i_1, \cdots, i_n, i_{n+1})$;
Learning Epochs: E;
Maximum Diffusion Steps: t;
Schedule β: *Linear*;
Approximator: $f_\theta(\cdot)$;
Output:
Well-trained *Approximator*: $f_\theta(\cdot)$;
while $j < E$ **do**
$s = \lfloor s' \rfloor, s' \sim U(0, t)$; `// Diffusion Step Sampling`
$\mathbf{x}_s \leftarrow q(\mathbf{x}_s|\mathbf{x}_0)$; `// Diffusion`
$(\mathbf{z}_1, ..., \mathbf{z}_n)$ $\leftarrow (\mathbf{e}_1 + \lambda_1(\mathbf{x}_s + \mathbf{d}_s), ..., \mathbf{e}_n + \lambda_n(\mathbf{x}_s + \mathbf{d}_s))$;`// Distribution Representation`
$\hat{\mathbf{x}}_0 \leftarrow f_\theta([\mathbf{z}_1, ..., \mathbf{z}_n])$; `// Reconstruction`
parameter update: $\mathcal{L}_{CE}(\hat{\mathbf{x}}_0, i_{n+1})$;
$j = j + 1$;
end while

where a, b are hyperparameters, which control the range of β. If $\beta_s > \tau$, $\beta_s = 0.1\beta_s$. τ is a truncated threshold, we set $\tau = 1$. 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 $q(\mathbf{x}_s|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_s; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1 - \overline{\alpha}_s)\mathbf{I})$. 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.,* $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$), we generate $\mathbf{x}_s$ as follows:

$$\mathbf{x}_s = \sqrt{\overline{\alpha}_s}\mathbf{x}_0 + \sqrt{1 - \overline{\alpha}_s}\epsilon \tag{15}$$

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 reversion 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:

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

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

where $\tilde{\mu}_t(\mathbf{x}_t, \hat{\mathbf{x}}_0) = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}\hat{\mathbf{x}}_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$, 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 is illustrated in Algorithm 1.

4.5 Loss Function and Rounding

The approximator requires the step embedding for reconstruction (ref. Equatoin 13). In diffusion phase, we sample a step index from a uniform distribution over range $[1, t]$. The corresponding step embedding is fed into the approximiator for model learning.

The objective function of diffusion models could be simplified as a mean-squared error (ref. Equation 8). However, we argue it may be not appropriate in recommendation scenarios for two reasons. First, rather than a continuous distribution, the item embedding is static in the latent space. Second, it is more ubiquitous to calculate the relevance between two vectors in terms of inner product for sequential recommendation. Consequently, in diffusion phase, we utilize cross-entropy loss for model optimization as follows:

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

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

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 $\mathbf{x}_0$ into the discrete item space for recommendation. The following rounding function is used:

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

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

5 EXPERIMENTS

We conduct 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 efficient is the model's training and inference?

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%

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 *Toys*[1] 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*[2] is a common benchmark dataset that includes one million movie ratings. *Steam*[3] is collected from a large online video game distribution platform. The dataset contains rich information (spanning October 2010 to January 2018) *e.g.,* users' play hours, price, category, media score, and developer information.

Following the common data preporcessing [20, 32, 38], 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. We adopt leave-one-out strategy for performance evaluation. To be specific, for all datasets, given a sequence $\mathcal{S} = \{i_1, i_2, ..., i_n\}$, we use the most recent interaction (i_n) for testing, the penultimate interaction (i_{n-1}) for validation, and the earlier ones ($\{i_1, i_2, ..., i_{n-2}\}$) for 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 can see 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 [21], 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[4] [14] utilizes GRU to model the sequential behavior of users for recommendation. **Caser**[5] [34] devises horizontal and vertical CNN to exploit user's recent sub-sequence behaviors for recommendation. **SASRec**[6] [20] utilizes a single-direction Transformer to model the implicit correlations of items and it is a competitive benchmark in sequential recommendation. **BERT4Rec**[7] [32] believes the unidirectional 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[8] [2] adopts attention mechanism and dynamic routing for user's multi-interest extraction and recommendation. **TiMiRec**[9] [38] is the most uptodate 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[10] [29] is a pioneer work that combines GRU and variational autoencoder for next item prediction. **ACVAE**[11] [42] utilizes the Adversarial Variational Bayes (AVB) framework to generate high-quality latent variable representations for recommendation. **STOSA**[12] [7] develops the vanilla self-attention as a Wasserstein self-attention to model the item representations and also inject uncertainty into model training via a stochastic Gaussian distribution for sequential recommendation.

5.3 Implementation Details

For a fair comparison, we followed [20, 32, 48] and utilized the Adam optimizer with the initial learning rate of 0.001. We initialized the parameters of Transformer based on a Xaiver normalization distribution and set the block numbers as 4. We fixed both the embedding dimension and hidden state size as 128 and batch size as 512. The dropout rate of turning off neurons is 0.1 for all datasets. 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 $\{2, 8, 32, 64, 128, 512\}$ (more discussion in Section 5.8). 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 [23, 25, 25] 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 performance of our proposed DiffuRec and the other methods is presented in Table 2.

DiffuRec consistently outperforms all baselines across the four datasets in terms of all six metrics. In particular, compared with the

[1] https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/
[2] http://files.grouplens.org/datasets/movielens/ml-1m.zip
[3] https://steam.internet.byu.edu/
[4] https://github.com/hidasib/GRU4Rec
[5] https://github.com/graytowne/caser
[6] https://github.com/kang205/SASRec
[7] https://github.com/FeiSun/BERT4Rec
[8] https://github.com/THUDM/ComiRec
[9] https://github.com/THUwangcy/ReChorus/tree/CIKM22
[10] https://github.com/noveens/svae_cf
[11] https://github.com/ACVAE/ACVAE-PyTorch
[12] 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	DiffuRec	▲%
Beauty	HR@5	1.0112	1.6188	3.2688	2.1326	2.0495	1.9044	0.9943	2.4672	3.5457	**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	9.5939	**11.1098***	15.80%
	NDCG@5	0.6084	0.9758	2.3989	1.3207	1.0503	1.2438	0.6702	1.6858	2.5554	**4.0047***	56.72%
	NDCG@10	0.9029	1.3602	3.2305	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	6.9393	**7.4587***	7.48%
	HR@20	3.1827	2.9500	9.2263	4.5889	6.9357	2.7156	1.9239	4.4061	9.5096	**9.8417***	3.49%
	NDCG@5	0.6983	0.5707	3.0105	1.1630	1.1571	0.7051	0.5580	1.5604	3.1017	**4.1667***	34.34%
	NDCG@10	0.9396	0.8510	3.7533	1.4870	1.7953	0.9123	0.7063	1.8452	3.8806	**4.7724***	22.98%
	NDCG@20	1.2724	1.1293	4.3323	1.9038	2.4631	1.1374	0.8446	2.1814	4.3789	**5.3684***	22.60%
Movielens-1M	HR@5	5.1139	7.1401	9.3812	13.6393	6.1073	16.2176	1.4869	12.7167	7.0495	**17.9659***	10.78%
	HR@10	10.1664	13.3792	16.8941	20.5675	12.0406	23.7142	2.7189	19.9313	14.3941	**26.2647***	10.76%
	HR@20	18.6995	22.5507	28.318	29.9479	21.0094	33.2293	5.0326	28.9722	24.9871	**36.7870***	10.71%
	NDCG@5	3.0529	4.1550	5.3165	8.8922	3.5214	10.8796	0.9587	8.2287	3.7174	**12.1150***	11.36%
	NDCG@10	4.6754	6.1400	7.7277	11.1251	5.4076	13.3059	1.2302	10.5417	6.0771	**14.7909***	11.16%
	NDCG@20	6.8228	8.4304	10.5946	13.4763	7.6502	15.7019	1.8251	12.8210	8.7241	**17.4386***	11.06%
Steam	HR@5	3.0124	3.6053	4.7428	4.7391	2.2872	6.0155	3.2384	5.5825	4.8546	**6.6742***	10.95%
	HR@10	5.4257	6.4940	8.3763	7.9448	5.4358	9.6697	5.8275	9.2783	8.5870	**10.7520***	11.19%
	HR@20	9.2319	10.9633	13.6060	12.7332	10.3663	14.8884	7.9753	14.4846	14.1107	**16.6507***	11.84%
	NDCG@5	1.8293	2.1586	2.8842	2.9708	1.0965	3.8721	1.8836	3.5429	2.9220	**4.2902***	10.80%
	NDCG@10	2.6033	3.0846	4.0489	4.0002	2.1053	5.0446	2.6881	4.7290	4.1191	**5.5981***	10.97%
	NDCG@20	3.5572	4.2073	5.3630	5.2027	3.3434	6.3569	3.2323	6.0374	5.5072	**7.0810***	11.39%

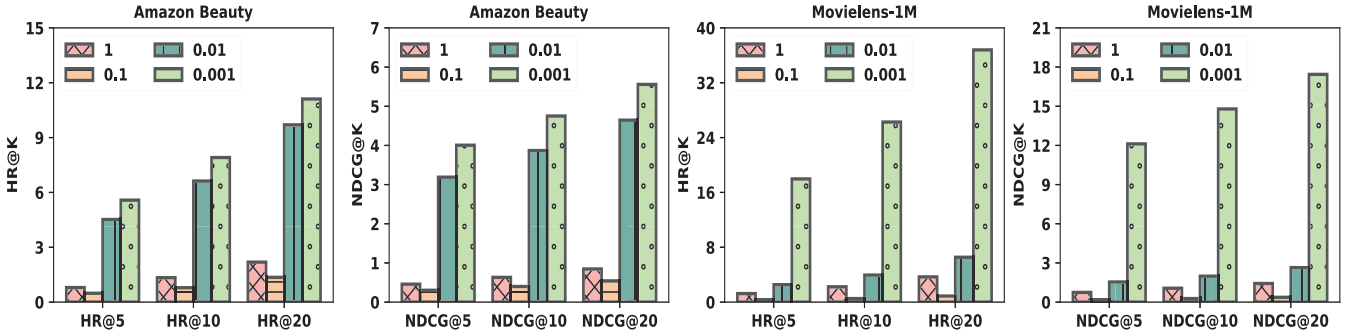

Figure 3: Parameter sensitivity of λ.

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 *Beauty*, *Toys*, *Movielens-1M* and *Steam* datasets, respectively. The significant performance gain suggests that DiffuRec can effectively accommodate the four characteristics (*i.e.,* multiple latent aspects, multiple interests, 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 overwhelm 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*. Additionally, our results also manifest that the incorporation of target item may facilitate the sequential pattern learning, especially for long sequences.

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 the 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. In general, the results of ACVAE and STOSA across all the datasets 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 *Approximater* and obtain the representation derived at the last step as $\hat{\mathbf{x}}_0$.
- **w R***: replaces the inner product in Equation 20 with the following variant as in [9]:

$$\frac{2\mathbf{x}_0 \cdot \mathbf{e}_i^T}{||\mathbf{x}_0||||\mathbf{e}_i||} - \left(\frac{\mathbf{x}_0}{||\mathbf{x}_0||} + \frac{\mathbf{e}_i}{||\mathbf{e}_i||}\right) \tag{21}$$

- **w Cosine**: replaces the *truncated linear* schedule with *cosine* schedule as in [15].
- **w L**: replaces the *truncated linear* schedule with *linear* schedule as in [25].
- **w Sqrt**:replaces the *truncated linear* schedule with *sqrt-root* schedule as in [23].

Table 3 shows the ablation results on *Beauty* and *Movielens-1M* 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* dataset while the space and time complexity drop significantly. And this phenomenon is also observed in CV domain that compared with other well-designed 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 [9] as an alternative for target item prediction, but *HR*/*NDCG* reduce by at least 68.71%/74.61% and 75.08%/75.93% on *Beauty* and *Movielens-1M* datasets, respectively. We believe it is because inner product operation is aligned with objective loss and may be more suitable for sequential recommendation [2, 20, 32]. Furthermore, as [12] emphasizes that different schedules will have some impact on the final result but could not acquire huge margin fluctuations. Our experiments also manifest this conclusion. Moreover,

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	3.1773	4.6685	6.9000	2.2253	2.7075	3.2682
	w R*	1.3036	2.2902	3.4768	0.7964	1.1130	1.4108
	w Cosine	5.2502	7.2967	10.2858	3.7908	4.4461	5.1974
	w L	5.1420	7.3428	10.2650	3.7538	4.4610	5.1965
	w Sqrt	5.1543	7.2871	10.4156	3.7600	4.4443	5.2288
	DiffuRec	**5.5758**	**7.9068**	**11.1098**	**4.0047**	**4.7494**	**5.5566**
Movielens-1M	w GRU	16.6016	24.6094	34.9772	10.9553	13.5348	16.1403
	w R*	4.0690	6.0872	9.1688	2.7787	3.4297	4.1978
	w Cosine	**18.1912**	26.4865	36.5072	**12.3441**	**15.0205**	**17.5560**
	w L	17.6270	26.3075	**37.1691**	11.9917	14.7935	17.5321
	w Sqrt	18.1392	**26.4931**	36.6742	12.2589	14.9559	17.5295
	DiffuRec	17.9659	26.2647	36.7870	12.1150	14.7909	17.4386

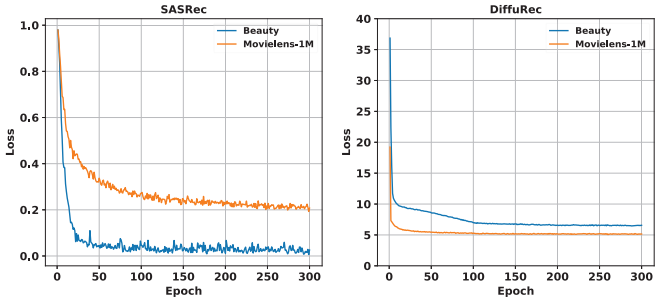

Figure 4: Curve of training loss on *Amazon-beauty* and *Movielens-1M* datasets

following [15, 23], we explore the mean-squared error loss and ϵ prediction via *Approximator* for target item representation reconstruction (*i.e.,* instead of predicted the target item representation $\hat{\mathbf{x}}$ straightforwardly. Specifically, we count on predicted ϵ for target item representation reconstruction, $\hat{\mathbf{x}}_0 = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x_t} - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon)$), 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 to the performance of DiffuRec. Figure 3 presents the performance patterns by varying λ values. We can see that a small λ generally resulted in better performance on both *Beauty* and *Movielens-1M* datasets, while when we increase the λ to 0.1, the *HR* and *NDCG* drop sharply, especially on *Movielens-1M* dataset. We suppose that a large λ may add too much noise to the historical interaction sequences, which will corrupt the original information and hinder the model to precisely understanding the user's interest.

5.7 Convergence and Efficiency (RQ4)

DiffuRec is a new paradigm for sequential recommendation which is also an iterative process in model training and inference. We thereby analyze its convergence and efficiency. All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU.

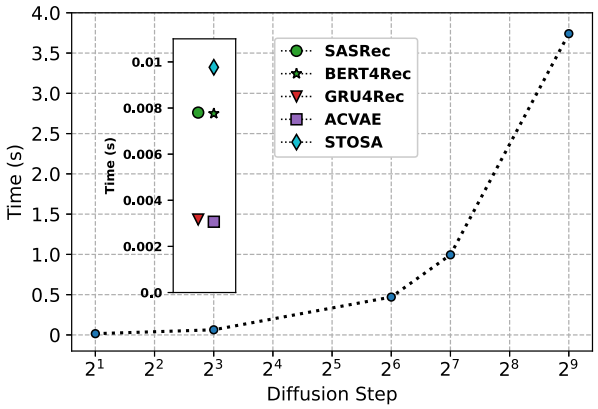

Figure 5: Inference time (in secs for one sample) on *Amazon-beauty*. The points in the inset box present the inference time of baselines. The folding line chart reveals the inference time for the different reverse steps.

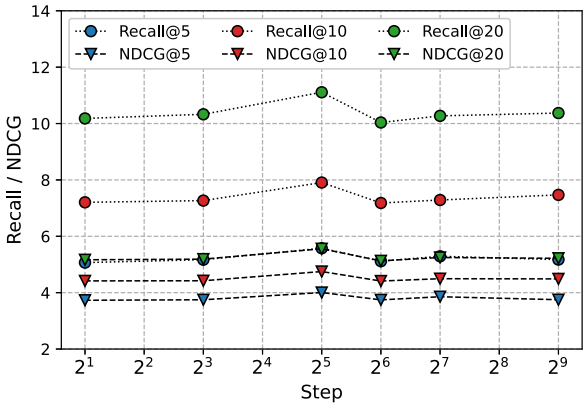

Figure 6: Recall and NDCG at different steps on *Amazon-beauty*.

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, hidden size, multi-head, and Transformer block numbers as the same, *i.e.,* Adam, 128, 4, and 4 respectively.

Figure 4 presents the loss curves on training process. Observe that on *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 *Beuaty* dataset (8.53), SASRec could not capture the long-term dependency 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 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. The results are shown in Figure 5.

Observing Figure 5, 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. Besides, with the increase of reverse step, the inference time for one sample rises exponentially, while the performance does not exhibit the same trend (observing Figure 6, the *HR* and *NDCG* do not improve obviously as we add diffusion and reversion steps). Balancing the quality of representation and the time, we consider a moderate diffusion step (*e.g.,* 32) is sufficient for SR. Furthermore, compared with Transformer-based methods, both GRU4Rec and ACVAE apply GRU as the backbone for recommendation. Hence, the inference time of them is similar and shorter than Transformer-based solutions. STOSA improves Transformer as a stochastic model and also adopts the Wassertein distance as an alternative to the inner product for discrepancy measure between two items. Consequently, the inference time is slightly longer than the unmodified Transformer-based methods.

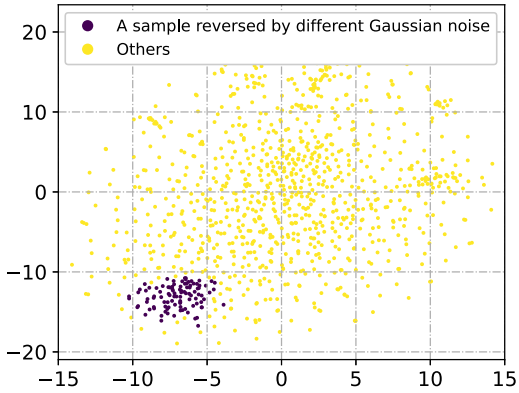

Figure 7: A t-SNE plot of the reconstructed $\mathbf{x}_0$. The purple points represent the same item but reversed by 100 different Gaussian noise for a specific historical sequence. The yellow points are the others each of which is reversed from a random sequence in *Beauty* dataset.

5.8 Visualization of Uncertainty

As we emphasize that attributed to the theoretical underpinnings of Diffusion framework, DiffuRec could inject some uncertainty in diffusion and reverse process, which is beneficial to the final performance. We then analyze the uncertainty of the reconstructed target item representation $\mathbf{x}_0$ reversed by our method. Figure 7 plots a set of reversed $\mathbf{x}_0$ vectors via t-SNE [36].

Observing Figure 7, we find that the reversed $\mathbf{x}_0$ from random sequences are dispersed throughout the whole space uniformly. 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 of existing solutions are deterministic

for a specific sequence. The merit of this uncertainty is much desired for the retrieval stage. Specifically, by reversing 100 different $\mathbf{x}_0$ for the exemplar sequence, the number of unique items that are ranked in their top-20 reaches 643. That is, we can generate different reversed $\mathbf{x}_0$ with parallel computing, consequently leading to the diversified retrieval 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 of adapting the powerful Diffusion model remain under-explored. We hope this work may shed light along this direction.

REFERENCES

[1] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. 2020. Learning gradient fields for shape generation. In *ECCV*. Springer, 364–381.

[2] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *KDD*. 2942–2951.

[3] 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).

[4] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.

[5] 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.

[6] Ziwei Fan, Zhiwei Liu, Shen Wang, Lei Zheng, and Philip S Yu. 2021. Modeling Sequences as Distributions with Uncertainty for Sequential Recommendation. In *CIKM*. 3019–3023.

[7] 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 *WWW*. 2036–2047.

[8] 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 *CIKM*. 433–442.

[9] Shansan 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).

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[11] 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).

[12] 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).

[13] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*. 843–852.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 33 (2020), 6840–6851.

[16] 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.

[17] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. 2022. Video diffusion models. *arXiv preprint arXiv:2204.03458* (2022).

[18] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax Flows and Multinomial Diffusion: Towards Non-Autoregressive Language Models. *CoRR* abs/2102.05379 (2021). arXiv:2102.05379

[19] 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.

[20] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[21] Walid Krichene and Steffen Rendle. 2022. On sampled metrics for item recommendation. *Commun. ACM* 65, 7 (2022), 75–83.

[22] 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 *CIKM*. 2615–2623.

[23] 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).

[24] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.

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

[26] 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).

[27] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

[29] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. 2019. Sequential variational autoencoders for collaborative filtering. In *WSDM*. 600–608.

[30] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).

[31] 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.

[32] 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 *CIKM*. 1441–1450.

[33] 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 *AAAI*, Vol. 35. 4384–4392.

[34] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[35] 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 *SIGIR*. 1632–1641.

[36] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS* 30 (2017).

[38] 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 *CIKM*. 2007–2016.

[39] 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 *SIGIR*. 209–218.

[40] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *RecSys*. 328–337.

[41] Teng Xiao, Shangsong Liang, and Zaiqiao Meng. 2019. Hierarchical neural variational model for personalized sequential recommendation. In *WWW*. 3377–3383.

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

[43] 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 *WWW*. 303–313.

[44] 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 *WWW*. 2216–2226.
[45] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. 2012. Auralist: introducing serendipity into music recommendation. In *WSDM*. 13–22.
[46] Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. Infovae: Balancing learning and inference in variational autoencoders. In *AAAI*, Vol. 33. 5885–5892.
[47] 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 *KDD*. 1059–1068.
[48] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is all you need for sequential recommendation. In *WWW*. 2388–2399.
[49] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2013. Using temporal data for making recommendations. *arXiv preprint arXiv:1301.2320* (2013).