Time Matters: Enhancing Sequential Recommendations with Time-Guided Graph Neural ODEs

Haoyan Fu*
haoyan-fu@bit.edu.cn
Beijing Institute of Technology
Beijing, China

Zhida Qin†
zanderqin@bit.edu.cn
Beijing Institute of Technology
Beijing, China

Shixiao Yang
ysx144_51@bit.edu.cn
Beijing Institute of Technology
Beijing, China

Haoyao Zhang
zhanghaoyao@bit.edu.cn
Beijing Institute of Technology
Beijing, China

Bin Lu
robinlu1209@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Shuang Li
shuangliai@buaa.edu.cn
Beihang University
Beijing, China

Tianyu Huang
huangtianyu@bit.edu.cn
Beijing Institute of Technology
Beijing, China

John C.S. Lui
cslui@cse.cuhk.edu.hk
The Chinese University of Hong Kong
Hong Kong, China

Abstract

Sequential recommendation (SR) is widely deployed in e-commerce platforms, streaming services, etc., revealing significant potential to enhance user experience. The core of SR lies in exploring the sequential relationships in historical user-item interactions. However, existing methods often overlook two critical factors: *irregular user interests* between interactions and *highly uneven item distributions* over time. The former factor implies that actual user preferences are not always continuous, and long-term historical interactions may not be relevant to current purchasing behavior. Therefore, relying only on these historical interactions for recommendations may result in a lack of user interest at the target time. The latter factor, characterized by peaks and valleys in interaction frequency, may result from seasonal trends, special events, or promotions. These externally driven distributions may not align with individual user interests, leading to inaccurate recommendations. To address these deficiencies, we propose TGODE to both enhance and capture the long-term historical interactions. Specifically, we first construct the user time graph and item evolution graph, which utilize user personalized preferences and global item distribution information, respectively. To tackle the temporal sparsity caused by irregular user interactions, we design a time-guided diffusion generator to automatically obtain an augmented time-aware user graph. Additionally, we devise a user interest truncation factor to efficiently identify sparse time intervals and achieve balanced preference inference. After that, the augmented user graph and item graph are fed into a generalized graph neural ordinary differential equation (ODE) to align with the evolution of user preferences and item distributions. This allows two patterns of information evolution to be matched over time. Experimental results demonstrate that TGODE outperforms baseline methods across five datasets, with improvements ranging from 10% to 46%. The code is available at https://github.com/Qin-lab-code/TGODE.

CCS Concepts

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

Keywords

Sequential Recommendation; Graph Neural Networks; Neural Ordinary Differential Equation; Diffusion Model

ACM Reference Format:
Haoyan Fu, Zhida Qin, Shixiao Yang, Haoyao Zhang, Bin Lu, Shuang Li, Tianyu Huang, and John C.S. Lui. 2025. Time Matters: Enhancing Sequential Recommendations with Time-Guided Graph Neural ODEs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737156

1 Introduction

Recommender systems, as the method of delivering personalized preference recommendations to users, effectively address the issue of information overload in the modern Internet era. Furthermore, many research efforts [4, 17, 23] have recognized that user preference information is partially embedded within dynamic sequential item access behaviors, which exhibit strong correlations with preference information. Consequently, the task of sequence recommendation (SR) can efficiently utilize these correlations to explicitly model user sequential behavior.

Numerous approaches have been proposed for SR tasks to capture users' sequential interaction histories. Early efforts employ

*Also with Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).
†Corresponding authors.

KDD '25, Toronto, ON, Canada.

ACM ISBN 979-8-4007-1454-2/25/08
https://doi.org/10.1145/3711896.3737156

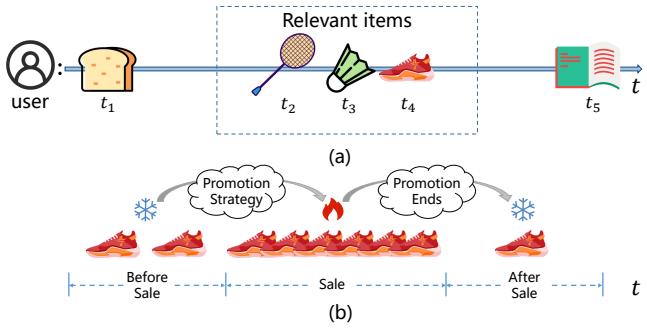


Figure 1: Illustration example of sequential recommendation with (a) irregular user interests and (b) highly uneven item distributions.

traditional methods, such as Markov chains [27] and RNNs [9, 26], to explore long-term sequence dependencies. With the success of transformers, a new wave of self-attentive SR models [11, 37, 40] is developed. However, these models often face limitations due to the inherent data sparsity in recommendation systems. Recently, Stochastic Differential Equation (SDE)-based probabilistic models have gained traction as a means to enhance existing SR methods. Diffusion models, such as DiffRec [32] and DreamRec [36], have emerged as promising tools for modeling complex interaction generation. Additionally, neural ODE [7, 25] has been applied within the recommendation domain to model continuous sequential interactions, further advancing the capabilities of SR systems.

Existing sequential recommendation methods, while effective in capturing sequential relationships in user sequences, ignore two critical factors: (i) **Irregular user interests.** General sequential modeling assumes regular user interactions over time, which is inconsistent with real-world recommender systems. As shown in Figure 1, user interactions follow an irregular distribution. Users may purchase closely related items consecutively in short time intervals when the sequential relationship of items represents the current user preferences. Comparatively, items outside of long time intervals are irrelevant to the current purchase behavior. Therefore, capturing irrelevant sequential patterns in a straightforward manner can have a negative effect. (ii) **Highly uneven item distributions.** Referring to Figure 1, the popularity of an item can change over time due to external factors. For example, the sales of an item may be highly increased due to a promotional campaign, and this effect is independent of the user's personal preferences. That is, the user's current behavior may be influenced by external promotions rather than following personal preferences. However, such outside influences over time cannot be mined by historical interactions, which limits the ability to make accurate recommendations.

Despite their great importance, the above two phenomena indicate that user interests and item distributions are both highly time-dependent with irregular intervals. Hence, incorporating this information into the SR system is a non-trivial problem. There are two challenges to deal effectively with both of these types of information. The first challenge is to alleviate the time-imbalanced user interactions. User behaviors tend to be highly concentrated in a short period of time and are absent in most timestamps, which exhibits the temporal sparsity issue. Most existing works mainly utilize the augmentation methods (such as random dropouts, similar sequences clustering) to supplement the interactions between cold start users and items from those similar ones with enough interactions. However, due to heterogeneous dynamics of user preference on time dimension, simply copying those interactions from similar ones and ignoring the time-aware interests may lead to severe user preference deviation and reduced performance. The second challenge lies in the mismatch between the distribution of personalized user interests and target items over time. As we demonstrate in the data analysis (See Section 2.2), the item distribution over time is mainly driven by some external factors, i.e., the promotions or advertising, which is independent of the evolution of user preferences. Failing to consider such phenomenon will make users crowded by disliked items and inevitably result in suboptimal recommendation results. Hence, it is necessary to characterize the evolution processes of both user individual preference and item distributions to efficiently align them and improve recommendation accuracy.

To address the above challenges, we propose a novel sequential recommendation framework with a **T**ime-**G**uided graph neural **ODE**s, named **TGODE**. First, we construct two tailored graphs, i.e. user time graph and item evolution graph, to capture individual user preference changes and explore dynamic item distributions respectively. Second, we designed a time-guided diffusion generator to augment the user time graph. Specifically, we apply a VAE encoder to compress interactions into latent vectors, and the output of another ODE module is co-coded to provide additional sequential information. We then perform a novel temporal embedding encoder to guide the diffusion process. Guided by temporal embedding, the diffusion model is able to recover user interactions corresponding to the time. Further, we propose a user preference inference module to carry out the generation process. Accordingly, we can generate potential user interests and achieve temporally balanced user preferences. Then, we propose a generalized graph neural ODE for matching critical evolutionary information in both the augmented user graph and original item graph. Specifically, our graph neural ODE could align the evolutionary patterns of two graphs over time. In this way, user interest transitions are matched with item trends on the timeline. In our framework, the outputs of these two modules serve as mutual inputs, and an iterative training strategy is employed to enhance their performance. We conduct extensive experiments on five real-world datasets and validate that our TGODE outperforms multiple state-of-the-art baselines.

Our contributions can be summarized as follows:

- We propose TGODE, a novel sequential recommendation framework that captures irregular user interests and highly uneven item distributions using two tailored graphs: a user time graph and an item evolution graph.
- We develop a time-guided diffusion generator to augment the user time graph, enabling the model to recover and generate user interactions that are temporally balanced and reflective of potential user preferences.
- Our generalized graph neural ODE aligns the temporal evolution of user interests with item trends, ensuring consistent and accurate recommendations over time.

- Extensive experiments on five real-world datasets show that TGODE outperforms state-of-the-art baselines, confirming its effectiveness in sequential recommendation tasks.

2 PRELIMINARY

2.1 Problem Definition

The traditional approach to sequence recommendation involves predicting the next item in a user interaction sequence s, denoted as $s = \{(v_1, t_1), (v_2, t_2), \ldots, (v_n, t_n)\}$, where n represents the length of sequence s, and $(v, t) \in s$ signifies an interaction between the sequence and item v at time t. Typically, each item v is initiated as its corresponding embedding $\mathbf{x}$. The objective is to forecast the $(n+1)$-th potential interaction given the first n interacting items and the target time t_{n+1}.

2.2 Data Analysis

In order to thoroughly investigate the widespread issues of irregular user interests and highly uneven item distributions in real-world scenarios, we conduct an extensive data analysis. For this study, we use two datasets from Amazon: Beauty and Toys. These are subsets of the extensive Amazon Review Data, and detailed statistics for both datasets are provided in the Appendix A.3. For simplicity, we present the data analysis results for the beauty dataset in the main text, while the results for the toys dataset are provided in the Appendix A.4.

2.2.1 Analysis of Irregular User Interests. A time interval represents the arbitrary time difference between two consecutive interactions for a user, reflecting the temporal connection between these interactions. We quantify the proportions of all pre-existing time intervals across different ranges in both datasets, as shown in Figure 2(a). Our observations reveal that, although nearly half of the interactions occur instantaneously, there were still numerous delayed interactions. The proportion of interactions across various time intervals is significant, indicating substantial temporal irregularity for the user. Almost 15% of the delayed interactions occur after more than 350 time intervals. These extremely long time intervals suggest that such interactions may only be weakly related to previous interactions. However, previous sequential modeling techniques often fail to distinguish these delayed interactions from those occurring instantaneously, potentially leading to biased interpretations of user preferences.

To further explore the relationship between interactions and time intervals for individual users, we randomly select three representative users and visualize their interaction timelines in different colors, as shown in Figure 2(b). The figure demonstrates that these users' interactions are clustered within certain time ranges and irregularly distributed at other timestamps. This irregular distribution underscores the inadequacy of purely sequential models in capturing time interval-related personalized interests, highlighting the necessity of considering irregular time intervals in modeling.

2.2.2 Analysis of Highly Uneven Item Distributions. In real-world scenarios, user interests evolve over time, and items themselves are subject to changes in distributions due to exposure, promotions, and other factors. To quantify this, we analyze the changes in item interactions over time. We divide the timeline into 250-day length

(a) Proportions of Time Intervals (b) User Interaction Timelines

(c) Item Emergence Ratios (d) Item Interaction Timelines

Figure 2: Data analysis regarding the Beauty dataset.

slices and calculate the proportion of an item's interactions within each slice to assess the item's concentrated emergence over time. In fact, the higher the proportion of emergence, the more concentrated the distribution of the item on the timeline is around a specific time, that is, the more popular the item is during a certain period. We categorize these proportions into the ranges of {0, 0-0.5, 0.5-0.75, 0.75-1, 1}. As illustrated in Figure 2(c), over 75% of items exhibit an emergence ratio of more than 75%, while items with an emergence ratio of less than 50% account for less than 10% of all items. This analysis demonstrates that items are highly concentrated within time slices, which we refer to as highly uneven item distributions.

To visualize item popularity more effectively, we select three items and display their interaction timelines, as shown in Figure 2(d). These visualizations confirm that items interact with numerous users around specific timestamps, corroborating the findings in Figure 2(c). However, this prevalent highly uneven item distribution is often overlooked and cannot be adequately captured by traditional time-series methods. Therefore, it is crucial to model data in a time-aware manner to accurately reflect these dynamics.

3 Method

3.1 Continuous Time Evolution Process

To capture the nuanced dynamics of user preferences over time, it is imperative to model continuous item interactions effectively. To this end, we devise a continuous time process, illustrated in Figure 3. Beginning with a collection of all user sequences, $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$, we integrate these sequences along a unified timeline, aligning their temporal occurrences.

However, it is clear from the data analysis that the unbalanced time intervals appear on both the individual user sequences and the overall item distribution, so it is necessary to construct the two graph structures separately to mine the corresponding information. Concretely, we construct two pivotal graphical structures:

- User Time Graph ($\mathcal{G}_{us}$). This graph captures individual user preferences. It is defined as $\mathcal{G}_{us} = \{\mathcal{V}_{us}, \mathcal{E}_{us}\}$. Here, $\mathcal{V}_{us} =$

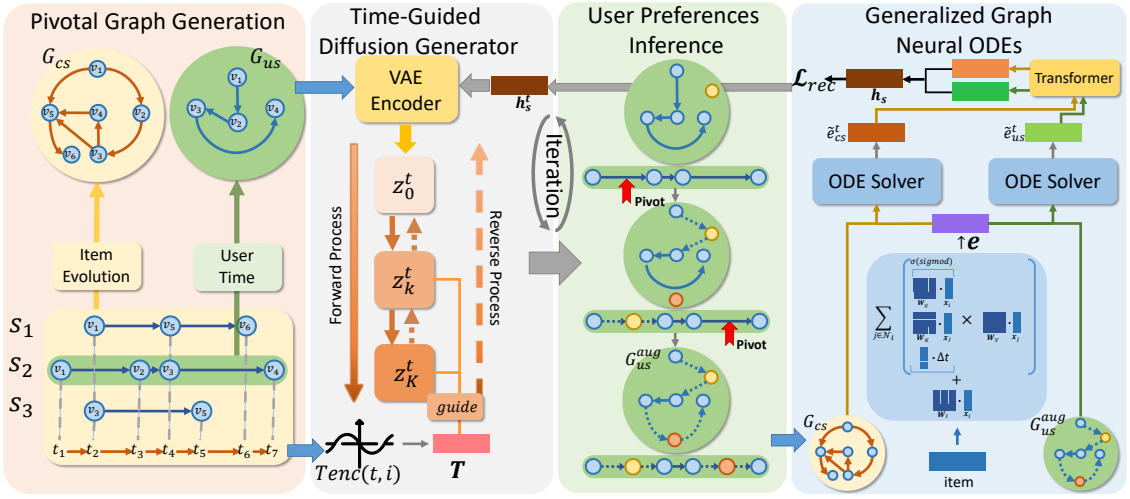


Figure 3: The overall architecture of our proposed method.

$\{v_i | v_i \in s\}$ represents all nodes corresponding to items within each user sequence s, while $\mathcal{E}_{us} = \{(v_i, v_j, t) | v_i, v_j \in s\}$ denotes the set of transitional relationships between two items within a sequence, labeled with their corresponding interaction times.

- Item Evolution Graph ($\mathcal{G}_{cs}$). This graph delineates the overarching temporal representation of items across all sequences. It is structured as $\mathcal{G}_{cs} = \{\mathcal{V}_{cs}, \mathcal{E}_{cs}\}$. In this case, $\mathcal{V}_{cs} = \mathcal{I}$ includes the entirety of items, while $\mathcal{E}_{cs} = \{(v_i, v_j, t) | v_i, v_j \in \mathcal{S}\}$ represents interactions occurring at time t across the entire sequence collection $\mathcal{S}$.

The distinctive focus of these graphs lies in their respective emphasis on individual user preferences and global item distributions. In particular, the graph $\mathcal{G}_{us}$ that models user preferences is more representative of current user interests. However, the number of interactions of the user graph $\mathcal{G}_{us}$, which only considers the sequence of individual behaviors, is much smaller than that of the evolution graph $\mathcal{G}_{cs}$. This leads to severe sparsity in the time dimension. Therefore, it is necessary to design a time-guided user preferences generator to fill in the missing potential user interactions. In this generative pattern, the potential user interest transition process can be obtained between long intervals of interactions.

3.2 Time-Guided Graph Neural ODEs

To efficiently extract temporal information from the two constructed graphs, we design a time-guided graph neural ODE model. The model consists of three parts: 1) a time-guided diffusion generator to augment user time graphs; 2) a user preferences inference to generate potential user interests; and 3) a generalized graph neural ODE to match evolution information on both graphs.

3.2.1 *Time-Guided Diffusion Generator.*

Due to irregular user interests, user time graphs are often sparse. Therefore, a generator is needed to address this sparsity in the temporal dimension. Inspired by the ability of diffusion models to retain detailed information during generation, we propose a novel time-specific approach. Specifically, we design a time-variant diffusion module to generate user behavior interactions over extended time intervals. Our approach focuses on creating potential transition paths for interacting items, thereby enhancing continuous user interest preferences.

To achieve this, we employ a time-guided denoising probabilistic model to learn time-specific user interaction information. We get the potential presentations with a vector encoder. Then we progressively disrupt the potential vectors and use the encoded temporal embedding to recover the original interaction records corresponding to the time. After training the model, a user preference inference strategy is utilized to generate an augmented user graph. This strategy effectively addresses the sparsity of user interactions over time and appropriately augments the missing edges in the original graph.

The Latent Vectors Encoder. First, we need to model the initial inputs $\mathbf{z}_0^t$ in order to perform the forward and backward processes. However, the user time graph only contains the current user's interactions, which is not sufficient for reasoning about potential interactions at other times. Inspired by the latent features encoder in latent diffusion [28], we use Variational Autoencoder (VAE) to compress graph structure and sequence information into the potential space at time t:

$$\mathbf{z}_0^t = \text{VAE}(\mathbf{A}_{us}^t, \mathbf{h}_s^t), \tag{1}$$

where $\mathbf{A}_{us}^t$ is the adjacency matrix of the graph $\mathcal{G}_{us}$ at time t and $\mathbf{h}_s^t$ is the sequence representation modeled by the subsequent graph neural ODE process. It is worth noting that the sequence information should match the current time t and be able to recover information from other moments. Our ODE process can model the representation of continuous time, thus helping the graph generation at other moments.

The Temporal Embedding Encoder. In the original user graph, each item node has a defined interaction time. To accurately capture the timing of user interactions, we transform timestamps into corresponding time embeddings. We design a temporal embedding

encoder to derive the temporal guidance for the diffusion model. Specifically, we combine sine and cosine values of varying frequencies with nonlinear transformations to encode time as follows:

$$\mathbf{c}_t = \text{Concat}\left(\sin\left(2\pi\omega_i t + b_i\right), \cos\left(2\pi\omega_i t + b_i\right)\right), \tag{2}$$

where ω_i is the first i frequency and b_i is the offset term.

Time-Guided Diffusion Process. Using the encoded temporal embedding, the diffusion model meticulously carries out the process of user graph corruption and reconstruction through forward and reverse processes. Specifically, we progressively disrupt the initial state $\mathbf{z}_0^t$ through K steps, known as the forward process. Following DDPM [10], we parameterize the forward process from $\boldsymbol{z}_0^t$ to $\boldsymbol{z}_K^t$ as follows:

$$\begin{aligned} q(\boldsymbol{z}_K^t|\boldsymbol{z}_0^t) &= \mathcal{N}(\boldsymbol{z}_K^t; \sqrt{\bar{\alpha}_K^t}\boldsymbol{z}_{K-1}^t, (1-\bar{\alpha}_K)\boldsymbol{I}); \\ \boldsymbol{z}_K^t &= \sqrt{\bar{\alpha}_K}\boldsymbol{z}_0^t + \sqrt{1-\bar{\alpha}_K}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \end{aligned} \tag{3}$$

where $\alpha_K = 1 - \beta_K, \bar{\alpha}_K = \prod_{k=1}^{K} \alpha_k$, for $1 - \bar{\alpha}_K \propto K$.

After the forward process corrupts the original graph structure into Gaussian noise, we control the reverse process by the learned time embedding $\mathbf{c}_t$. This process starts with $\boldsymbol{z}_K^t$ and gradually removes noise through a neural network to restore the user graph:

$$p_\theta(\boldsymbol{z}_{k-1}^t|\boldsymbol{z}_k^t, \mathbf{c}_t) = \mathcal{N}(\boldsymbol{z}_{k-1}^t; \boldsymbol{\mu}_\theta(\boldsymbol{z}_k^t, \mathbf{c}_t, k), \Sigma_\theta(\boldsymbol{z}_k^t, \mathbf{c}_t, k)), \tag{4}$$

where the terms $\boldsymbol{\mu}_\theta(\boldsymbol{z}_k^t, \mathbf{c}_t, k)$ and $\Sigma_\theta(\boldsymbol{z}_k^t, \mathbf{c}_t, k)$ represent Gaussian parameters generated by neural networks.

Time Smoothness Optimization. To optimize the parameter θ within the neural network, we maximize the Evidence Lower Bound (ELBO) associated with the initial state $\boldsymbol{z}_0^t$. After derivation of the formula, the loss function for step k is as follows:

$$\begin{aligned} \mathcal{L}_k &= \mathbb{E}_{q(\boldsymbol{z}_k^t|\boldsymbol{z}_0^t)}\left[D_{KL}(q(\boldsymbol{z}_{k-1}^t|\boldsymbol{z}_k^t, \boldsymbol{z}_0^t)) \parallel p_\theta(\boldsymbol{z}_{k-1}^t|\boldsymbol{z}_k^t, \mathbf{c}))\right] \\ &= \mathbb{E}_{q(\boldsymbol{z}_k^t|\boldsymbol{z}_0^t)}\left[\frac{1}{2}\left(\frac{\bar{\alpha}_{k-1}}{1-\bar{\alpha}_{k-1}} - \frac{\bar{\alpha}_k}{1-\bar{\alpha}_k}\right)\|\hat{\boldsymbol{z}}_\theta(\boldsymbol{z}_k^t, \mathbf{c}, k) - \boldsymbol{z}_0^t\|_2^2\right] + C, \end{aligned} \tag{5}$$

where $\hat{\boldsymbol{z}}_\theta(\boldsymbol{z}_k^t, \mathbf{c}, k)$ is obtained by feeding the input vector $\boldsymbol{z}_k^t$ and the time embedding of step k into a MLP to predict $\boldsymbol{z}_0^t$.

Considering Eq. (5), we can optimize the ELBO by utilizing $\sum_{k=1}^{K} \mathcal{L}_k$. In practical implementation, we uniformly sample step $k \sim \mathcal{U}(1, K)$ for optimizing $\mathcal{L}(\boldsymbol{z}_0^t, \theta)$, formalized as follows:

$$\mathcal{L}(\boldsymbol{z}_0^t, \theta) = \mathbb{E}_{k\sim\mathcal{U}(1,K)}\mathcal{L}_k. \tag{6}$$

Although this optimization ensures that the restored $\hat{\boldsymbol{z}}_0^t$ is as close as possible to the original state $\boldsymbol{z}_0^t$, it may lead to over-smoothing of time. Since most of the original item interactions are within the same time period, the model tends to generate nodes within repeated timestamps. This makes it difficult for the model to learn user behavior under sparse timestamps, which is not conducive to generating high-quality samples. To mitigate this issue, we regularize the edge weights to increase the irregularity of the graph indirectly. The overall loss function of our time-guided diffusion module is:

$$\mathcal{L}_{diff} = \mathcal{L}(\boldsymbol{z}_0^t, \theta) + \sum_{i,j}\left|\hat{\boldsymbol{z}}_0^t\right|. \tag{7}$$

3.2.2 *User Preferences Inference.*

After training a time-guided diffusion model, a key question is how to infer the potential interest of users at other times. On the timeline, there are continuous interactions during some time periods, while others may be almost absent. We devise a criterion, called the user interest truncation factor, to identify time periods where user interest is missing.

User Interest Truncation Factor. Specifically, given an overall timeline T, we divide it into m time pivots $t^p = \{t_1^p, t_2^p, \ldots, t_m^p\}$. Assuming the original time series $t^o = \{t_1^o, t_2^o, \ldots, t_n^o\}$, for each timestamp t_i^o, we define the function $g(t_i^o)$ to find the closest time pivot t_j^p:

$$g(t_i^o) = \arg\min_{t_j^p \in t^p}\left|t_j^p - t_i^o\right|. \tag{8}$$

Based on the above function, we can get the set of uncovered pivots $t_{set}^p = \{t \in t^p \mid t \notin g(t_i^o)\}$. This set contains the time pivots where the user's interest is missing. Next we calculate the user interest truncation factor, which indicates the number of interactions that should be generated on each missing time pivot:

$$l_{num} = max(1, \frac{|t^o|}{|t_{set}^p|}). \tag{9}$$

Augmented User graphs Inference Process. Afterwards, we sample through the trained generator by utilizing the latent factor to infer potential user preferences. We proceed with K sampling steps and inference to derive the augmented adjacency matrix $\hat{\boldsymbol{z}}_0^t = \boldsymbol{\mu}_\theta(\boldsymbol{z}_K^t, \mathbf{c}_t, K)$ at a given time t. The augmented adjacency matrix predicts potential interactions within the user graph. Note that, as proven in previous work [13, 32], the inference step is optimal when K is set to 0, in this case $\boldsymbol{z}_K^t = \boldsymbol{z}_0^t$. With this setting, the process is considered as denoising the sparse recommendation data.

With the set of uncovered pivots t_{set}^p and the user interest truncation factor l_{num} obtained from the above calculation, we identify the top l_{num} highest scoring probabilistic interaction edges. By merging these edge sets with the original graph, we form the augmented graph $\mathcal{G}_{aug}$, which can be expressed as

$$\mathcal{G}_{us}^{aug} = \mathcal{G} \cup \text{topK}(\mathcal{G}_{us}^{t_1}) \cup \text{topK}(\mathcal{G}_{us}^{t_2}) \cup \ldots \cup \text{topK}(\mathcal{G}_{us}^{t_m}). \tag{10}$$

Through user preference truncation factors, we infer the potential user interests in the missing temporal pivots. With this inference process, we obtain enhanced user time graphs, which enhance the continuous dynamic interests of users on the timeline.

3.2.3 *Generalized Graph Neural ODEs.*

The time-guided diffusion model interpolates the edges of user graph structures with extreme temporal imbalances to maintain the presence of nodes at pivot times. With this strategy, we enhance the intrinsic factors that influence user preferences. However, in long-term time user interest is simultaneously influenced by objective extrinsic factors. In this case, user interest changes may not match the dynamic item distributions. To capture the temporal consistency of users and items during evolution, we propose a generalized graph neural ODE model. It can model time-synchronous user and item dynamics.

Time Sensitive Latent State Encoder. To efficiently capture temporal information, we propose an attention-based GNN as our graph

encoder. Specifically, given the initialized item embedding $\mathbf{x}$ and two graphs $\{\mathcal{G}_{us}^{aug}, \mathcal{G}_{cs}\}$, the item representations of both graphs are encoded as follows:

$$\alpha_{ij} = \text{Sigmod}\left(\mathbf{a}^\top \left[\mathbf{W}_Q\mathbf{x}_i, \mathbf{W}_K\mathbf{x}_j, \Phi(t)\right]\right),$$
$$\mathbf{e}_{us} = \sum_{j\in\mathcal{N}_i^{us}} \alpha_{ij}\mathbf{W}_V\mathbf{x}_j + \mathbf{W}_l\mathbf{x}_i,\ \mathbf{e}_{cs} = \sum_{j\in\mathcal{N}_i^{cs}} \alpha_{ij}\mathbf{W}_V\mathbf{x}_j + \mathbf{W}_l\mathbf{x}_i, \tag{11}$$

where α_{ij} represents the attention weight of node i towards node j, $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$, and $\mathbf{W}_l$ denote the weight matrices, $\Phi(t)$ signifies the time position embedding, and $\mathcal{N}_i$ denotes the set of neighbors of node i. Through the application of a graph encoder, the initialized item embedding $\mathbf{x}$ is transformed into two item representations $\mathbf{e}_{us}$ and $\mathbf{e}_{cs}$, which tend to focus on the changes in the users' interests and the evolution of item distributions, respectively.

Generalized Graph ODE Solver Function. To accurately align with the dynamic evolution of item representations in continuous time, we propose a temporal graph ODE solver function = $\mathbf{f}_\theta$ to simulate the derivative in the ODE process. To predict the representation of items at any given moment, we encode the timestamps and corresponding graph structures in the ODE solver function, where $\mathbf{f}_\theta = \mathbf{f}_\theta\{\mathbf{e}_{us}(t), \mathbf{e}_{cs}(t), \mathbf{g}(t), \mathcal{G}_{us}^{aug}(t), \mathcal{G}_{cs}(t)\}$ and $\mathbf{g}$ denotes the timestamp encoding function.

Upon obtaining the precise time embedding $\mathbf{g}(t)$ through position encoding, we align the item representations of the user and item graphs at time t by means of the ODE function $\mathbf{f}_\theta$. Given the adjacency matrix $\mathbf{A}_{us} \in \mathcal{G}_{us}^{aug}$ and the corresponding degree matrix $\mathbf{D}_{us}$, we opt for the normalized adjacency matrix $\tilde{\mathbf{A}}_{us}$ to facilitate neighborhood information transfer among graph nodes. Notably, as time progresses within the ODE, multiple layers are stacked, and normalization proves effective in mitigating potential gradient explosion issues. Given the directed nature of both graph types, we employ normalization via $\mathbf{D}_{us}^{-1}\mathbf{A}_{us}$ instead of $\mathbf{D}_{us}^{-\frac{1}{2}}\mathbf{A}_{us}\mathbf{D}_{us}^{-\frac{1}{2}}$. The ODE solver function is defined as follows:

$$\frac{d\tilde{\mathbf{e}}^t}{dt} = f_{us}(\mathbf{e}_{us}, g(t)) + f_{cs}(\mathbf{e}_{cs}, g(t)),\ f_{cs} : \tilde{\mathbf{e}}_{cs}^t = \mathbf{W}_a[\mathbf{e}_{cs}^t, \mathbf{g}(t)]$$
$$f_{us} : \tilde{\mathbf{e}}_{us}^{l+1(t)} = \mathbf{W}_b[\mathbf{e}_{us}^{l(t)}, \mathbf{g}(t)],\ \mathbf{e}_{us}^{l+1(t)} = \sigma\left(\tilde{A}_{us}^t\mathbf{e}_{us}^{l(t)}\mathbf{W}_c\right) + \mathbf{e}_{us}^{l(t)} \tag{12}$$

where $\mathbf{W}_a$, $\mathbf{W}_b$, $\mathbf{W}_c$ denote trainable parameter matrices, σ represents the activation function, l indicates the number of layers, and $\tilde{\mathbf{A}}_{us}^t$ signifies the normalized adjacency matrix of the subgraph $\mathcal{G}_{us,<t}^{aug}$. The derivatives of item representations at t timestamps are modeled by combining two networks f_{us} and f_{cs}. As the user time graph is augmented by a time-guided diffusion generator, we meticulously mine higher-order neighborhood relationships in the f_{us}. In contrast, the item evolution graph contains the global distribution of items, and we use the MLP to explicitly model the influence of external factors within the f_{cs}. During the ODE evolution, both factors simultaneously affect the item state at the corresponding moment. Ultimately, we use the Runge-Kutta method to obtain the final item representation $\tilde{\mathbf{e}}_{us}^{t_{n+1}}$ and $\tilde{\mathbf{e}}_{cs}^{t_{n+1}}$ at the target time t_{n+1}.

3.3 Model Prediction and Optimization

By evolving the states of both graphs through a neural ODE process, we are able to obtain the target time item representations $\tilde{\mathbf{e}}_{us}^{t_{n+1}}$ and $\tilde{\mathbf{e}}_{cs}^{t_{n+1}}$ at the target time. The final item vectors are then fed through a transformer decoder [31], and obtain the final sequence representation:

$$\mathbf{h}_s = ||Decoder(\tilde{\mathbf{e}}_{us}^{t_{n+1}})|| + ||Decoder(\tilde{\mathbf{e}}_{cs}^{t_{n+1}})||. \tag{13}$$

where $||\cdot||$ is the L_2 norm.

To predict the probability of user sequence interaction with a specific item v, we compute the dot product between users' sequence embedding $\mathbf{h}_s$ and the target item embedding $\tilde{\mathbf{e}}_v$, in order to assess their correlation:

$$\hat{\mathbf{y}} = \text{softmax}(\|\mathbf{h}_s\|^\top \|\tilde{\mathbf{e}}_v\|). \tag{14}$$

We employ the cross-entropy loss function to calculate the optimized objective for the primary recommendation task, the definition is as follows:

$$\mathcal{L}_{rec} = -\sum_{v=1}^{|V|} \mathbf{y}\log(\hat{\mathbf{y}}) + (1-\mathbf{y})\log(1-\hat{\mathbf{y}}). \tag{15}$$

Note that we alternately train the two modules, time-guided diffusion generation and generalized graph neural ODEs, to iteratively improve the recommendation performance. The specific training process is shown in the Appendix A.1. In addition, we provide a discussion 3.4 that further elaborates on the effectiveness of our iterative update strategy.

3.4 Discussion

In this section, we examine the effectiveness and necessity of our iterative updating strategy.

First, directly applying a diffusion model is not feasible. Our task requires predicting user preferences at a target time, which involves understanding continuous time in long sequences. However, as user interests tend to drift over time, diffusion models struggle to capture accurate distributions over extended periods, as evidenced by the poor performance of two diffusion models in Table 1. This observation highlights the need to integrate representations with accurate temporal information into the diffusion process.

Second, a continuous-time representation using a graph neural ODE is essential. Although our generator is trained on existing timestamps, it must also reason about representations at other times. Neural ODEs can model node representations at arbitrary time points, providing a foundation for the diffusion model to learn feature distributions beyond the training timestamps.

Finally, the diffusion model and the graph neural ODE are complementary. The diffusion generator enhances the graph neural ODE by supplying augmented graphs that capture detailed changes in user interests. While the neural ODE offers continuous-time representations that help the diffusion generator reason about distributions at any time. Together, these components reinforce each other, leading to improved overall performance.

4 EXPERIMENTS

We conduct extensive experiments to verify the performance of our TGODE model by answering the following questions:

- **RQ1:** How does the performance of our TGODE compare to different types of recommendation methods in competitive scenarios?

- **RQ2:** What is the impact of each specific key module within our TGODE framework on the overall performance?
- **RQ3:** How does our method perform with different sequence decoders?
- **RQ4:** How does TGODE perform in handling datasets with different levels of sparsity?
- **RQ5:** How interpretable is TGODE in real situations?

4.1 Experiment Settings

4.1.1 Dataset and Evaluation. To verify the validity of our approach, we conduct extensive experiments on five public datasets: (a) Beauty, (b) Sports, (c) Toys, (d) Video, and (e) ML-100k, which are widely used in the SR tasks. The first four datasets come from the Amazon platform and ML-100k is obtained from the MovieLens. Unlike general SR tasks, we employ a distinct evaluation method that is more consistent with prediction under the target time. More details are shown in the Appendix A.3.

4.1.2 Baselines and Evaluation Metrics. We evaluate our TGODE with different types of baseline comparisons: the traditional methods (NARM [16], SRGNN [35], GRU4REC [9]); the transformer methods (SASRec [15], SSE-PT [34], CORE [11], MAERec [37]); the diffusion methods (DreamRec, DiffRec); and the continues time methods (TisasRec [17], GNG-ODE [7], GDERec [25]). We adopt three widely used evaluation metrics in recommendation: Recall@k (R@k, k=5, 10, 20), MRR@k (M@k, k=5, 10, 20) and NDCG@k (N@k, k=5, 10, 20).

4.1.3 Experiment Details. We implement our proposed method using PyTorch and optimize the parameters with the Adam optimizer. Since the original user graph is too sparse, we set the step K to 5 in the training phase and set the step K to 0 in the generation process referring to [13, 32]. The hyperparameters for which we performed the grid search are shown below: the learning rate is sampled between 1e-1 and 1e-5. The range of the number of layers of the graph neural network is {1, 2, 3, 4, 5} and the number of heads of the Transformer ranges from 1 to 3. The dimension of embeddings varies in the range of {32, 64, 128, 256}. After we search for the optimal hyperparameters, we compare the results with other baseline models under the same settings.

4.2 Comparison of Performance (RQ1)

In this section, we conduct a comprehensive performance evaluation of TGODE and the aforementioned baselines. For simplicity, here we present the experimental results for Recall@k and NDCG@k, while the results for MRR@k can be found in the Appendix A.5. Table 1 presents the experimental results of each model on three datasets, yielding the following observations:

The recommendation models utilizing Transformers, such as SASRec, SSE-PT, CORE, and MAERec, demonstrate superior performance on multiple datasets compared to traditional models like NARM, SRGNN, and GRU4Rec. These observations highlight the practicality of utilizing Transformers for modeling sequential information and their adaptability to various recommendation tasks in different contexts.

Models that consider continuous time, such as TisasRec, GNG-ODE, and GDERec, have a competitive advantage over the two types of models mentioned above. This emphasizes the importance of capturing the dynamically changing aspects of user preferences. However, they ignore the temporal sparsity of user interests. As a result, the user preferences captured by these models change in time in jumps, leading to biased recommendation results.

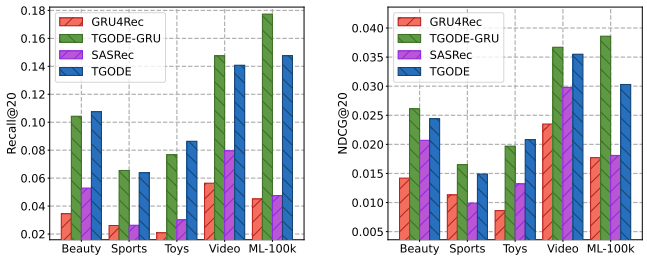


Figure 4: Comparison with different sequence decoders.

Approaches based on diffusion models, such as DreamRec and DiffRec, perform poorly on SR tasks. In short-term historical interactions, diffusion models are able to generate appropriate recommended items with their denoising ability. However, under long-term historical interactions, users' interests change dynamically. At this point, the diffusion model may generate previously preferred items that do not match the current user preferences. Therefore, it is inappropriate to simply apply the diffusion model in long sequence recommendation.

Our approach consistently outperforms the baselines on multiple metrics, effectively addressing the challenges of dynamic user preferences and uneven item distribution. We mine temporally-guided diffusion models for temporal sequences of user interaction information and use user preference inference to generate latent user interests under missing temporal pivots. Afterwards, the graph neural ODE model matches the consistency of user interests and item distributions over the evolutionary process, effectively improving the recommendation performance.

4.3 Ablation Experiment (RQ2)

The ablation experiment section compares three model variants to assess the impact of each module in TGODE on performance.

- The **base** variant removes all valid modules.
- The **w/o Diff** variant removes the diffusion generator module.
- The **w/o ODE** variant removes the generalized graph neural ODEs module.
- The **w/o cs** variant removes the item evolution graph.

Table 2 shows the performance comparison of our original model and its four ablated variants across two evaluation metrics. Based on the ablation results, the following observations can be analyzed and obtained:

From the significant performance degradation observed in the variant "base" compared to the original model, we can observe that merely utilizing a Transformer for sequence encoding in recommendation tasks is insufficient. This variant ignores irregular user interests and highly uneven item distributions, leading to great performance degradation.

From the experimental results of the variant "w/o Diff", it is evidenced for the efficacy of the diffusion model in reconstructing interaction sequences, consequently mitigating irregular user interests. With our designed time-guided strategy and user preference

Table 1: Performance comparison among TGODE and twelve baselines over five datasets.

Dataset	Metrics	Tranditional Method			Transformer Method				Diffusion Method		Continues Time Method			Ours	*improve.*
		NARM	SRGNN	GRU4REC	SASRec	SSE-PT	CORE	MAERec	DreamRec	DiffRec	TisasRec	GNG-ODE	GDERec	TGODE	
Beauty	R@5	0.0112	0.0015	0.0131	0.0223	0.0258	0.0263	0.0253	0.0005	0.0101	0.0145	0.0304	0.0151	**0.0422**	38.82%
	R@10	0.018	0.004	0.0223	0.0357	0.0443	0.0541	0.0473	0.001	0.0181	0.0255	0.05	0.0267	**0.0709**	31.05%
	R@20	0.0281	0.0083	0.0345	0.0529	0.0706	0.09	0.0796	0.0018	0.0305	0.0435	0.0775	0.0436	**0.1077**	19.67%
	N@5	0.0071	0.0007	0.0082	0.012	0.0154	0.0124	0.0149	0.0003	0.006	0.0085	0.0192	0.0093	**0.0243**	26.56%
	N@10	0.0093	0.0015	0.0111	0.0164	0.0214	0.0214	0.0219	0.0005	0.0085	0.012	0.0255	0.0131	**0.0336**	31.76%
	N@20	0.0118	0.0025	0.0142	0.0207	0.028	0.0305	0.03	0.0007	0.0117	0.0166	0.0325	0.0173	**0.0429**	32.00%
Sports	R@5	0.01	0.0112	0.0113	0.0101	0.0153	0.0164	0.0177	0.0003	0.0098	0.0099	0.0188	0.013	**0.0257**	36.70%
	R@10	0.0159	0.0176	0.0155	0.0163	0.0267	0.0323	0.0296	0.0006	0.0162	0.0161	0.0319	0.0211	**0.0426**	31.89%
	R@20	0.026	0.0282	0.0261	0.0263	0.0435	0.0552	0.0468	0.0009	0.0259	0.0254	0.0504	0.0348	**0.0664**	20.29%
	N@5	0.0067	0.0073	0.0073	0.0054	0.0095	0.0078	0.0114	0.0002	0.0059	0.0063	0.0119	0.0083	**0.0152**	27.73%
	N@10	0.0086	0.0094	0.0086	0.0074	0.0132	0.0129	0.0152	0.0002	0.008	0.0083	0.0161	0.0109	**0.0206**	27.95%
	N@20	0.0111	0.012	0.0113	0.0099	0.0174	0.0187	0.0196	0.0003	0.0104	0.0107	0.0208	0.0144	**0.0266**	27.88%
Toys	R@5	0.0082	0.004	0.0079	0.0168	0.0174	0.0283	0.0176	0.0007	0.0063	0.0127	0.0184	0.0071	**0.0392**	38.52%
	R@10	0.0107	0.0084	0.0123	0.0232	0.0297	0.047	0.0307	0.0012	0.0113	0.0186	0.0287	0.0131	**0.0620**	31.91%
	R@20	0.0149	0.0131	0.0209	0.0301	0.0456	0.0726	0.0497	0.002	0.0182	0.0262	0.0446	0.0226	**0.0869**	19.70%
	N@5	0.0056	0.0024	0.005	0.0094	0.0095	0.0162	0.0102	0.0004	0.0034	0.0074	0.0118	0.0042	**0.0221**	36.42%
	N@10	0.0064	0.0038	0.0064	0.0115	0.0135	0.0199	0.0144	0.0006	0.005	0.0093	0.0151	0.0061	**0.0291**	46.23%
	N@20	0.0075	0.005	0.0086	0.0132	0.0175	0.0254	0.0192	0.0008	0.0067	0.0113	0.0191	0.0085	**0.0354**	39.37%
Video	R@5	0.0145	0.014	0.0211	0.0301	0.0335	0.0338	0.0465	0.0004	0.0155	0.0217	0.0416	0.0186	**0.0565**	21.51%
	R@10	0.0247	0.0246	0.0354	0.0508	0.0602	0.0707	0.0775	0.0005	0.0271	0.0351	0.0678	0.0333	**0.0930**	20.00%
	R@20	0.0375	0.0342	0.0564	0.0795	0.0981	0.1224	0.1238	0.001	0.046	0.0559	0.107	0.0569	**0.1423**	14.94%
	N@5	0.0099	0.0096	0.0136	0.0159	0.0207	0.0169	0.0284	0.0004	0.0088	0.0139	0.0261	0.0116	**0.0341**	20.07%
	N@10	0.0132	0.0131	0.0182	0.0225	0.0293	0.0287	0.0383	0.0004	0.0125	0.0182	0.0346	0.0163	**0.0462**	20.63%
	N@20	0.0165	0.0154	0.0235	0.0298	0.0388	0.0418	0.0501	0.0005	0.0173	0.0235	0.0446	0.0222	**0.0588**	17.37%
ML-100K	R@5	0.0068	0.0077	0.0128	0.0144	0.022	0.0142	0.0206	0.0035	0.0035	0.0087	0.0388	0.013	**0.0433**	11.60%
	R@10	0.0133	0.0109	0.0242	0.0266	0.0436	0.0313	0.0399	0.0072	0.0091	0.0191	0.0684	0.0275	**0.0831**	21.49%
	R@20	0.0241	0.0208	0.0451	0.0475	0.0819	0.0637	0.0771	0.015	0.0228	0.0377	0.1188	0.0524	**0.1486**	25.08%
	N@5	0.0043	0.0046	0.0078	0.0085	0.0136	0.0074	0.012	0.0022	0.0021	0.0054	0.0245	0.0101	**0.0271**	10.61%
	N@10	0.0065	0.0056	0.0125	0.0125	0.0209	0.0133	0.0184	0.0034	0.0041	0.0089	0.0347	0.0155	**0.0413**	19.02%
	N@20	0.0094	0.0084	0.0177	0.0181	0.0312	0.022	0.0282	0.0054	0.0078	0.0138	0.0479	0.0226	**0.0586**	22.34%

Table 2: Ablation study.

Ablation	Beauty		Sports		Toys	
Settings	R@20	N@20	R@20	N@20	R@20	N@20
TGODE	**0.1077**	**0.0429**	**0.0664**	**0.0266**	**0.0869**	**0.0354**
base	0.0594	0.0235	0.0448	0.0184	0.0189	0.0077
w/o Diff	0.1008	0.0393	0.0609	0.0233	0.0846	0.0339
w/o ODE	0.0733	0.0294	0.0527	0.0220	0.0476	0.0191
w/o cs	0.1023	0.0400	0.0557	0.0224	0.0803	0.0323

inference, the generator is able to consciously fill in user interactions in irregular temporal pivots, thus enhancing continuous user interest changes.

From the experimental results of the variant "w/o ODE", we can detect that ODE effectively captures and aligns dynamic user preferences and item distributions. The performance variations across datasets highlight the significance of this variant, with a notable 45% performance decrease observed on the Toys dataset.

The experimental results of the variant "w/o cs" elucidate the impact of item distributions bias on the unbiased representation of the model. By neglecting the temporal popularity information of items, the model is unable to take into account changes and trends over time in items influenced by external factors. Specifically, the user's current purchasing behavior may simply be influenced by the most recently popular items, rather than his personal preferences.

4.4 Performance with Different Decoders (RQ3)

To examine the expressive capacity of different encoders for sequential representation and their impact on the overall model performance, we employ two encoding approaches: GRU and Transformer. Additionally, we apply GRU4Rec (GRU) and SASRec (Transformer) as a comparison.

Figure 4 illustrates the impact results of our model on two different sequence encoding modules related to two baselines that utilize the corresponding module. We derive two observations from the results. Firstly, SASRec outperforms GRU4Rec due to the utilization of the Transformer framework, which enables the modeling of global contextual structures. In contrast, GRU4Rec's performance is limited as it relies solely on modeling long-term temporal dependencies, and its encoding capacity is constrained by its relatively simple sequential structure. This is also why we outperform the GRU variant in Beauty and Toys data. Secondly, GRU4Rec achieves performance close to SASRec on the Sports and ML-100K datasets. This can be attributed to the fragmented nature of interactions in the datasets, where there is no pronounced sequential order but rather a certain degree of long-term dependency. Consequently, our variant of GRU exhibits better performance in such scenarios. Overall, both variants of our approach outperform the direct application of the two corresponding sequence decoders, demonstrating the advancement of our proposed model.

4.5 In-depth Exploration of Model (RQ4)

In real-world datasets, sequence length is indicative of the sustained preferences among diverse user populations. To explore the efficacy of our TGODE model in capturing sustained preferences across different interest groups, we partition the sequence lengths into three categories: <10, 10-20, and >20. We compare the performance of our TGODE model with two baseline models (SASRec and GNG-ODE) as shown in Figure 5.

From the illustration, we can obtain three key observations. Firstly, when sequences are short, user interactions tend to be

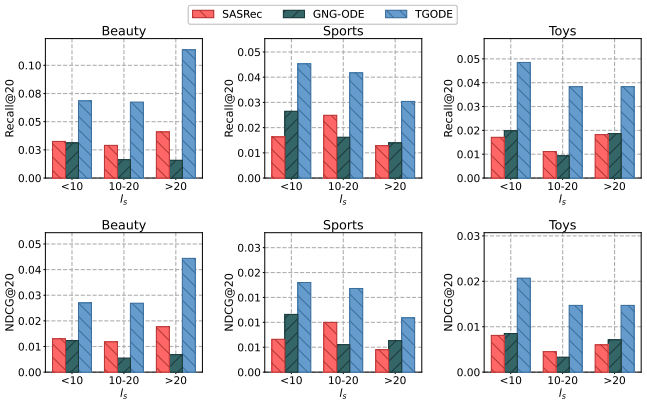


Figure 5: Sequence length within <10, 10-20, >20.

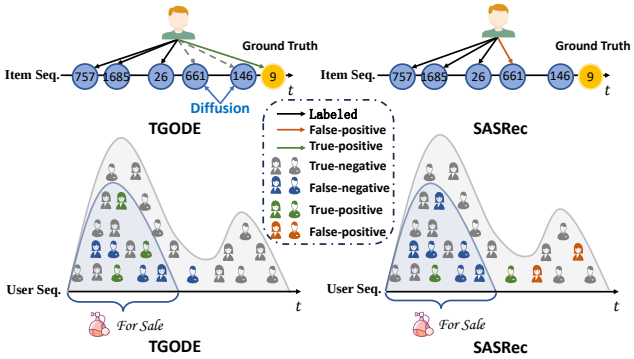


Figure 6: Case study of two examples on the Beauty dataset.

sparse with cold start scenarios, and there is little need for long-term memory or complex modeling of sequential dependencies as the contextual information is relatively limited. This allows the SASRec model, which utilizes the Transformer architecture, to more effectively model and utilize the limited contextual information when handling short sequences. Secondly, irregular time intervals within longer sequences will present challenges for the Transformer architecture in encoding long-range dependencies. In contrast, the GNG-ODE model excels at capturing such dependencies by modeling the continuous evolution process of sequences. Notably, in the Beauty dataset, SASRec outperforms GNG-ODE due to relatively shorter sequence lengths, which limit the manifestation of long-distance dependency relationships. Thirdly, our TGODE outperforms the two baselines across three variants of datasets with different sequence lengths. Our approach enhances temporally irregular user behavior, which enables capturing user preferences on sparse short sequences. At the same time, we match the patterns of evolution of user interests and item distributions over time, which allows us to improve the recommendation performance under processing long sequences.

4.6 Case Study (RQ5)

To validate TGODE's effectiveness, we extract two representative examples from the Beauty dataset. We also compare them with SASRec in terms of item and user sequences to demonstrate our approach's efficacy.

In Figure 6, we extract a user's interaction item sequence, where temporal gaps between items are represented as time intervals. To better characterize distinct sample categories, we utilize patterns of varying colors to represent them. Specifically, green patterns are employed to denote True-positive instances, while other color schemes are as annotated in the figure legend. Labeled data v_{757}, v_{6185}, and v_{26} are included, and the ground truth item to be predicted is v_9. Our TGODE effectively predicts item v_9 by incorporating item v_{146} through diffusion and capturing long-term temporal dependencies using ODE. In contrast, SASRec's lack of awareness of time interval information and long-distance dependencies results in an erroneous prediction of item v_{661} adjacent to the labeled data.

In the lower part of Figure 6, we present an example demonstrating the distribution of users' interactions with items over a specific period (shown in gray). We select a popular item (shown in blue) that receives loads of users' interactions during this period for illustration. Our TGODE capture the popularity of items, successfully predicts numerous user interactions during its peak popularity and decreases recommendations as its popularity declines. In contrast, SASRec lacks a sense of item popularity over time and continues recommending the item to users even when it is no longer popular.

5 Conclusion

In conclusion, this paper introduces TGODE, a novel framework that addresses the challenges of irregular user interests and uneven item distributions in sequential recommendation systems. By iteratively training the time-guided diffusion generator and generalized graph neural ODEs, TGODE captures dynamic user behaviors and shifting item trends. Extensive experiments on multiple real-world datasets confirm that TGODE significantly outperforms existing state-of-the-art methods, demonstrating its effectiveness in enhancing recommendation accuracy.

Acknowledgments

This work is supported by the National Nature Science Foundation of China under Grant Number U2441242. And in part by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), under Grant No. GML-KF-24-22.

References

[1] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 378–387.

[2] Liu Chong, Xiaoyang Liu, Rongqin Zheng, Lixin Zhang, Xiaobo Liang, Juntao Li, Lijun Wu, Min Zhang, and Leyu Lin. 2023. CT4Rec: Simple yet Effective Consistency Training for Sequential Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3901–3913.

[3] Xinyan Fan, Zheng Liu, Jianxun Lian, Wayne Xin Zhao, Xing Xie, and Ji-Rong Wen. 2021. Lighter and better: low-rank decomposed self-attention networks for next-item recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 1733–1737.

[4] 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.

[5] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 364–373.

[6] Penglei Gao, Xi Yang, Rui Zhang, Ping Guo, John Y Goulermas, and Kaizhu Huang. 2024. EgPDE-Net: Building Continuous Neural Networks for Time Series Prediction With Exogenous Variables. *IEEE Transactions on Cybernetics* (2024).

[7] Jiayan Guo, Peiyan Zhang, Chaozhuo Li, Xing Xie, Yan Zhang, and Sunghun Kim. 2022. Evolutionary preference learning via graph nested gru ode for session-based recommendation. In *Proceedings of the 31st ACM international conference on information & knowledge management*. 624–634.

[8] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 191–200.

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

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[11] Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao. 2022. Core: simple and effective session-based recommendation within consistent representation space. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1796–1801.

[12] Ziqi Huang, Kelvin CK Chan, Yuming Jiang, and Ziwei Liu. 2023. Collaborative diffusion for multi-modal face generation and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6080–6090.

[13] Yangqin Jiang, Yuhao Yang, Lianghao Xia, and Chao Huang. 2024. DiffKG: Knowledge Graph Diffusion Model for Recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 313–321.

[14] Ming Jin, Yu Zheng, Yuan-Fang Li, Siheng Chen, Bin Yang, and Shirui Pan. 2022. Multivariate time series forecasting with dynamic graph neural odes. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[16] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[17] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.

[18] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems* 35 (2022), 4328–4343.

[19] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1734–1743.

[20] Qidong Liu, Fan Yan, Xiangyu Zhao, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Feng Tian. 2023. Diffusion augmentation for sequential recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1576–1586.

[21] Cheng Lu, Kaiwen Zheng, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 2022. Maximum likelihood training for score-based diffusion odes by high order denoising score matching. In *International Conference on Machine Learning*. PMLR, 14429–14460.

[22] Shitong Luo and Wei Hu. 2021. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4583–4592.

[23] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory augmented graph neural networks for sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 5045–5052.

[24] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten De Rijke. 2020. Star graph neural networks for session-based recommendation. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 1195–1204.

[25] Yifang Qin, Wei Ju, Hongjun Wu, Xiao Luo, and Ming Zhang. 2024. Learning graph ODE for continuous-time sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[26] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *proceedings of the Eleventh ACM Conference on Recommender Systems*. 130–137.

[27] 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.

[28] 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.

[29] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*.

[30] 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.

[31] 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).

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

[33] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 169–178.

[34] 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.

[35] 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.

[36] Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. 2024. Generate What You Prefer: Reshaping Sequential Recommendation via Guided Diffusion. *Advances in Neural Information Processing Systems* 36 (2024).

[37] Yaowen Ye, Lianghao Xia, and Chao Huang. 2023. Graph Masked Autoencoder for Sequential Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 321–330.

[38] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 729–732.

[39] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3836–3847.

[40] Yipeng Zhang, Xin Wang, Hong Chen, and Wenwu Zhu. 2023. Adaptive disentangled transformer for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3434–3445.

[41] Weiheng Zhong, Hadi Meidani, and Jane Macfarlane. 2023. Attention-based Spatial-Temporal Graph Neural ODE for Traffic Prediction. *arXiv preprint arXiv:2305.00985* (2023).

A Appendix

A.1 Algorithm Details

The training process for our TGODE model is shown in Algorithm 1.

A.2 Efficiency Analysis

We analyze the time complexity of our two main modules: the diffusion generator and the graph neural ODE. For the time-guided diffusion generator, the complexity is $O(m(K_1 \times n^2 + K_2 \times n^2 \times d))$, where m is the number of generation, K_1 and K_2 are the steps of forward process and reverse process, n is the number of items, d is the hidden size. Unlike general diffusion models, we perform m-times generation in the inference process. Since the number of generation times can be manually tuned to balance performance and efficiency, the added time overhead is acceptable. Additionally, The complexity of ODE is $O(k \times (l \times E \times d + n \times d^2))$. k is the step size of ODE, l is the number of layers and E is the number of edges. Here, the number of edges is the primary factor affecting complexity. Overall, the time complexity of our modules is mainly determined by the graph size, but the computational overhead can be managed by tuning the relevant parameters as the graph size increases.

A.3 Dataset and Evaluation

A.3.1 Dataset Description. In this paper, we conduct experiments on five typical public datasets and compare TGODE with other baselines. This website (https://jmcauley.ucsd.edu/data/amazon/) contains all the datasets. A brief description of these five datasets is given below: **Beauty**, **Sports**, **Toys** and **Video** collect interaction

Algorithm 1 The Procedure of TGODE

Input: The sequences $\mathcal{S}$ and initial item embedding $\mathbf{x}$.

Initialization parameters θ_{diff} and θ_{rec};
Construct user time graph $\mathcal{G}_{us}$ and item evolution graph $\mathcal{G}_{cs}$;
for each iteration **do**
  for each batch **do**
    Get the sequence representation $\mathbf{h}_s^t$ under timestamp t from θ_{rec};
    Calculate the initial input $\mathbf{z}_0^t$ and time embedding $\mathbf{c}_t$ corresponding to the timestamp t according to Equation (1-2);
    Perform the Forward Process based on Equation (3) and the Reverse Process based on Equation (4);
    Calculate the loss function $\mathcal{L}_{diff}$ of the time-guided diffusion module based on Equation (5-7);
    Update parameters θ_{diff} to minimize $\mathcal{L}_{diff}$;
  end for
  Calculate the the set of uncovered pivots t_{set}^p and the user interest truncation factor l_{num} according to Equation (8-9);
  Generate l_{num} graphs on the uncovered pivots through the trained generator and merge them with the original user graph $\mathcal{G}_{us}$ into an augmented graph $\mathcal{G}_{us}^{aug}$ through Equation (10);
  for each batch **do**
    Calculate the item representations $\mathbf{e}_{us}$ and $\mathbf{e}_{cs}$ through the time sensitive latent state Encoder based on Equation (11);
    The final item representations $\tilde{\mathbf{e}}_{us}^{t_{n+1}}$ and $\tilde{\mathbf{e}}_{cs}^{t_{n+1}}$ are derived at the target time t_{n+1} through a generalized graph neural ODE solver function based on Equation (12);
    Obtain the representation of the sequence $\mathbf{h}_s$ through Equation (13) and compute the recommendation loss function $\mathcal{L}_{rec}$ based on Equation (14-15).
    Update parameters θ_{rec} to minimize $\mathcal{L}_{rec}$;
  end for
end for

Output: Optimized θ_{diff} and θ_{rec}.

Table 3: Statistics of datasets

Dataset	User	Item	Inter	Ave.Len	Density
Beauty	22363	12101	198502	8.87	99.92%
Sports	35598	18357	296337	8.32	99.95%
Toys	19412	11924	167597	8.63	99.92%
Video	24303	10672	231780	9.53	99.91%
ML-100k	943	1682	100000	106.04	93.69%

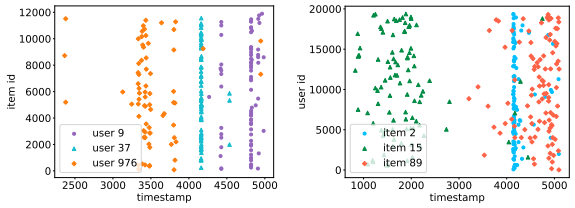
records between users and items in different fields on Amazon respectively. Users also have certain differences in their interaction preferences for items in different fields. Meanwhile, **ML-100k** is a stable benchmark dataset that tracks the ratings of 100k movies on MovieLens. The detailed information of these five datasets is shown in Table 3.

A.3.2 Evaluation. In a realistic scenario, the recommender system only has data that is prior to the current moment. Naturally, all training data should be earlier in time than the evaluation and test data. Traditional sequence evaluation uses the leave-one-out method to predict the last interaction of each user. In contrast, we divide all interactions in chronological order in a ratio of 8:1:1. With this division method, we consider all interactions as prediction targets. In this case, the input sequence is the sequence of corresponding user interactions previous to that target time.

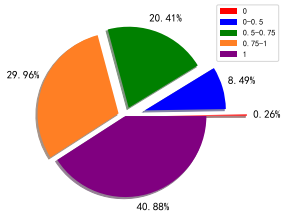
Similar to the approach in Section 2.2, we present the relevant analysis of the toys dataset in figure 7, including user interests, item emergence ratios, and item distributions.

As shown in figure 7(a), in the Toy dataset, the interactions of these users are also clustered within a certain time range, and exhibit irregular distribution at other timestamps. Additionally, as seen in figure 7(c), more than 80% of the items have an occurrence rate higher than 75%, while items with an occurrence rate below 10% account for less than 10%. Furthermore, as shown in figure 7(b), item interactions with numerous users are also concentrated at specific timestamps. These analytical results exhibit the same trend as the Beauty dataset in Section 2.2, further emphasizing the necessity of considering irregular time intervals and modeling in a time-aware manner.

A.4 Data Analysis for Toys



(a) User Interaction Timelines (b) Item Interaction Timelines



(c) Item Emergence Ratios

Figure 7: Data analysis regarding the Toys dataset.

A.5 Addition Comparison with Baselines

We also show how TGODE compares with other baselines in terms of MRR@k (k=5, 10, 20), and the results are shown in Table 4. As can be observed from the results, our model comprehensively outperforms the existing baseline on the MRR evaluation metric, demonstrating the superiority of our approach.

A.6 Parameter Analysis

To investigate the potential benefits of utilizing multiple propagation layers, we varied the number of layers in the GNN. Specifically, we conduct experiments on three different datasets, setting layer numbers in the range of $\{1, 2, 3, 4, 5\}$.

Table 4: Additional comparison results with the baselines on the MRR metric.

Dataset	Metrics	Tranditional Method			Transformer Method				Diffusion Method		Continues Time Method			Ours	*improve.*
		NARM	SRGNN	GRU4REC	SASRec	SSE-PT	CORE	MAERec	DreamRec	DiffRec	TisasRec	GNG-ODE	GDERec	TGODE	
Beauty	M@5	0.0057	0.0005	0.0066	0.0086	0.0119	0.008	0.0115	0.0003	0.0046	0.0066	0.0156	0.0075	**0.0181**	16.03%
	M@10	0.0066	0.0008	0.0078	0.0104	0.0144	0.0116	0.0143	0.0003	0.0057	0.008	0.0182	0.009	**0.0219**	20.33%
	M@20	0.0073	0.001	0.0086	0.0116	0.0162	0.0141	0.0165	0.0004	0.0065	0.0092	0.02	0.0102	**0.0224**	12.00%
Sports	M@5	0.0056	0.006	0.006	0.0039	0.0076	0.005	0.0094	0.0001	0.0046	0.0052	0.0097	0.0068	**0.0118**	21.65%
	M@10	0.0064	0.0069	0.0065	0.0047	0.0091	0.0071	0.0109	0.0002	0.0055	0.006	0.0114	0.0079	**0.0140**	22.81%
	M@20	0.0071	0.0076	0.0073	0.0053	0.0103	0.0087	0.0121	0.0002	0.0061	0.0066	0.0126	0.0088	**0.0156**	23.81%
Toys	M@5	0.0047	0.0018	0.0041	0.0069	0.0069	0.0075	0.0078	0.0003	0.0025	0.0057	0.0096	0.0033	**0.0164**	70.83%
	M@10	0.0051	0.0024	0.0047	0.0078	0.0085	0.0106	0.0095	0.0004	0.0031	0.0065	0.011	0.0041	**0.0192**	74.55%
	M@20	0.0054	0.0027	0.0052	0.0082	0.0096	0.0123	0.0108	0.0004	0.0036	0.007	0.012	0.0047	**0.0210**	75.00%
Video	M@5	0.0084	0.0081	0.0112	0.0112	0.0165	0.0114	0.0224	0.0003	0.0066	0.0114	0.021	0.0093	**0.0279**	24.55%
	M@10	0.0097	0.0095	0.013	0.0139	0.02	0.0163	0.0264	0.0003	0.0082	0.0131	0.0245	0.0112	**0.0318**	20.45%
	M@20	0.0106	0.0102	0.0145	0.0158	0.0225	0.0198	0.0296	0.0004	0.0095	0.0145	0.0271	0.0128	**0.352**	18.92%
ML-100K	M@5	0.0032	0.0036	0.0062	0.0063	0.01	0.0049	0.009	0.0017	0.0015	0.0039	0.0186	0.0063	**0.0209**	13.37%
	M@10	0.0041	0.004	0.0077	0.0079	0.0128	0.0071	0.0115	0.0022	0.0023	0.0053	0.0225	0.0082	**0.0267**	18.67%
	M@20	0.0048	0.0047	0.0091	0.0093	0.0154	0.0093	0.014	0.0027	0.0032	0.0065	0.0259	0.0099	**0.0305**	17.76%

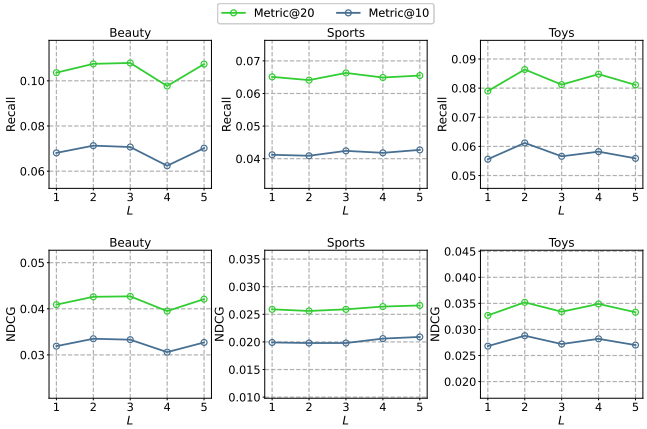


Figure 8: The impact of hyperparameter L on Beauty, Sports and Toys data.

The results are shown in the Figure 8. TGODE achieves optimal performance when the number of layers is set to 2 or 3 on the Beauty and Sports datasets. This is because as the number of layers increases, the model's ability to handle complex data improves, which leads the model to capture richer information. However, as the layer continues to increase, the performance of the model deteriorates. This is due to the phenomenon of over-smoothing that occurs when using excessively deep layers, which degrades the model's performance. Experimental results confirm that setting the number of layers to 2 yields satisfactory performance across three datasets.

A.7 Related Work

A.7.1 Sequential Recommendation. The purpose of SR is to formulate and predict users' sequential interaction behaviors, capturing temporal dependencies and patterns. Current methodologies for SR can be primarily categorized into markov chains [8, 27], RNN-based [9, 19, 26, 38], graph neural network (GNN)-based [1, 24, 33, 35], and Transformer-based [2–4, 11, 15, 17, 30] approaches. Early works adopt markov chains to consider long-term dependencies and capture item-item sequence dependencies in SR. With the proliferation of deep learning techniques, the widespread adoption of RNN and GNN technologies has greatly facilitated the modeling of SR tasks by effectively excavating the information inherent in sequential and graph structures. For example, SRGNN [35] utilizes graph attention networks to model individual sessions, enabling the representation of both the user's global preferences and current interests. The Transformer's multi-head attention mechanism and positional encoding enable it to directly and efficiently model global dependencies within sequences, thereby benefiting sequential recommendation tasks. MAERec [37] utilizes Transformer and graph-masked autoencoder to alleviate the problem of requiring high-quality graph embeddings in self-supervised learning.

A.7.2 Diffusion Models. The diffusion model has grown into a powerful deep learning model in recent years, and significant achievements have been made in related work in multiple aspects, either by optimizing model performance through practice [29] or by theoretically increasing model capacity [21]. In applications, diffusion models dominate multiple challenging interdisciplinary tasks, including computer vision [22, 39], natural language processing [18], multi-modal modeling [12], and other interdisciplinary applications. For the field of recommender systems, approaches utilizing diffusion models [20, 32, 36] have also achieved superior performance with their excellent denoising and generation capabilities. For example, DiffuASR [20] mitigates data sparsity and long-tail user issues in SR by employing a diffusion-based pseudo sequence generation framework and sequence U-Net model. DreamRec [36] utilizes guided diffusion to generate an ideal positive sample, eliminating negative samples and accurately capturing the user's real preferences.

A.7.3 Neural ODEs. Neural ODEs are powerful tools for handling dynamic system modeling and sequence modeling, with significant advantages in continuous modeling, flexible depth, and memory efficiency. Given its advantages in spatiotemporal sequences, ode has been widely applied in related aspects, e.g., time series prediction [6, 14] and traffic flow forecasting [5, 41]. In the field of recommendation systems, ODE's time modeling ability is also well suited to user interaction sequences. For graph recommendation, GDERec [25] employs two customized GNNs trained alternately in an autoregressive manner to model and predict the evolution of user preferences, enabling tracking of the underlying system's dynamics under irregular observations. While for SR, considering the continuity of dynamic user preferences, GNG-ODE [7] extends the idea of neural ODEs to continuous time session graphs and proposes a method to align the update time steps of time session graphs.