



ARTICLE

# CalendarRec: Multi-Order Temporal Context for Sequential Recommendation

Duseo Pak<sup>1</sup>, Jiho Kim<sup>2,\*</sup> and Hongchul Lee<sup>1,\*</sup>

<sup>1</sup>Department of Industrial and Management Engineering, Korea University, Seoul, Republic of Korea

<sup>2</sup>Software Education Institute, Chung-Ang University, Seoul, Republic of Korea

\*Corresponding Authors: Jiho Kim. Email: jihonav@cau.ac.kr; Hongchul Lee. Email: hclee@korea.ac.kr

Received: 14 May 2026; Accepted: 27 August 2026; Published: 22 October 2026

**ABSTRACT:** As the health functional food (HFF) market expands owing to increased interest in health and well-being globally, greater emphasis is placed on the importance of personalized recommendation systems within e-commerce platforms. Existing sequential recommendation models, such as recurrent neural networks and transformer-based models, face challenges of high computational complexity when processing long sequences and are unable to adequately reflect changes in user preferences over time. To address these problems, we propose CalendarRec, which uses a state space model-based Mamba structure to enhance computational efficiency and integrate temporal context information. The proposed model learns temporal patterns of user behavior by combining item click sequences with the corresponding month, day, and day of week information, which are embedded into independent latent spaces. In experiments using HFF click log data collected from an e-commerce site (GS SHOP), the proposed model outperformed existing models such as GRU4Rec, SASRec, BERT4Rec, and TiSASRec in all evaluation metrics. Specifically, it achieved relative improvements of 9.5% and 14.2% in terms of HR@10 and NDCG@10, respectively, over Mamba4Rec, a framework that applies the Mamba structure to recommendation systems. Despite a modest increase in computational cost relative to Mamba4Rec, the proposed model remains substantially more efficient than transformer-based approaches, confirming its suitability for real-time recommendation environments. This study suggests that temporal context information contributes substantially to improving recommendation quality in the evaluated HFF domain and additional experiments on public benchmark datasets further support the generalizability of the proposed approach to other domains.

**KEYWORDS:** E-commerce; health functional food; sequential recommendation; state space model; time context-aware

## 1 Introduction

Consumer engagement in online environments has grown explosively, driven by the rapid expansion of e-commerce platforms. Consumers have moved beyond simple purchasing activities, revealing their latent preferences through various interactions, such as product exploration and reviews. The accumulation of multidimensional interaction data allows firms to precisely analyze individual needs and enhance their services. Shifting from standardized offerings to providing hyper-personalized, data-driven services is becoming a critical competitive advantage in the contemporary e-commerce market. This phenomenon is particularly pronounced in the health functional food (HFF) market, where consumer needs are highly segmented. However, as the range of available products continues to expand, consumers increasingly struggle to find items suited to their individual needs. Accordingly, to support consumers' purchase decisions,

e-commerce platforms have widely adopted recommendation systems that analyze historical behavioral records and user preferences to predict items of interest.

Recommendation systems have continuously evolved to enhance accuracy and personalization. Early approaches were dominated by collaborative filtering and content-based filtering [1], which rely on static user-item interaction matrices. However, these methods have limitations in capturing sequential behavioral patterns and user interests that shift dynamically over time. In online shopping environments characterized by many products and users, it is essential to promptly and accurately identify real-time evolving behavioral patterns and suggest appropriate items to enhance user satisfaction and revenue [2]. Thus, recent research has focused on sequential recommendation systems (SRSs), which model shifts in user preferences over time. Architectures based on transformers and recurrent neural networks (RNNs), such as BERT4Rec [3], SASRec [4], and GRU4Rec [5], have been widely adopted with significant performance improvements. However, transformer-based models face bottlenecks owing to increased computational complexity and memory requirements when processing long sequences [6], and RNNs have been criticized for their inability to fully preserve long-term dependencies as sequence length increases and their prolonged training times [7]. The Mamba architecture, based on the state space model (SSM), is an alternative to address these issues. It has recently gained attention in the recommendation field for its ability to effectively learn long-term dependencies with linear time complexity. Furthermore, the purchasing decisions of users cannot be fully explained merely by chronological lists of past purchases. Beyond historical interests, various contextual factors such as the time of day, day of week, and month of product exploration, along with personal information and metadata, significantly influence purchasing behavior [8]. Therefore, an effective recommendation framework must capture both long-term dependencies and multidimensional temporal contexts beyond simple sequential histories.

To meet these demands, this study proposes CalendarRec, a model that integrates multidimensional temporal context based on Mamba4Rec [9], which applies the Mamba structure to recommendation systems. Compared to transformer-based models, Mamba4Rec enables efficient parallel processing while effectively capturing temporal information, making it advantageous for real-time recommendation and complex behavior prediction. Building upon the Mamba4Rec framework, we propose a recommendation model that enhances performance by integrating user behavior sequences and specific interaction time information of each item.

The contributions of this study are as follows:

1. We propose an SSM-based recommendation architecture that effectively integrates temporal information while enabling efficient inference for e-commerce service environments.
2. We validate the effectiveness and data reliability of the proposed model using a large-scale real-world dataset collected from an e-commerce environment.
3. Through comparative experiments and ablation studies, we empirically analyze the impact of integrating temporal information on recommendation performance within the HFF domain and present a more effective recommendation method.

The remainder of this paper is structured as follows. [Section 2](#) discusses the theoretical background and related work. [Section 3](#) delineates the architecture and algorithmic framework of the proposed model. [Section 4](#) evaluates the performance through experimental results, and [Section 5](#) summarizes the findings and suggests avenues for future research.

## 2 Related Work

Owing to the recent development of the Internet, users are struggling with information overload due to excessive exposure to online content. Therefore, most e-commerce services use recommendation systems that can selectively provide useful information to users. We introduce studies on SRS, which are primarily used in e-commerce services.

### 2.1 Sequential Recommendation Systems

In e-commerce, the three representative architectures applied to SRS are RNN-based, transformer-based, and recently, SSM-based Mamba models. A notable example of a model that uses RNN architecture is GRU4Rec [5], while representative transformer-based models include SASRec [4] and BERT4Rec [3]. Mamba4Rec [9] is a recommendation model that uses an SSM-based Mamba architecture.

Models based on sequential recommendations, such as GRU4Rec and SASRec, predict the next item based on the past interactions of a user; however, they cannot use bidirectional contextual information within the sequence, owing to their unidirectional structure. Accordingly, Sun et al. [3] proposed the BERT4Rec model, which applies the transformer bidirectional encoding structure to SRS. Based on BERT, BERT4Rec uses masked item prediction to learn bidirectional dependencies among items in a user behavior sequence while enabling parallel computation. However, owing to the structural characteristics of the transformer, the time complexity is  $O(L^2)$  for a sequence length  $L$ , which is a disadvantage: the inference time becomes much longer as the sequence length increases. Liu et al. [9] proposed Mamba4Rec, which reduces model inference time while improving recommendation performance, with an inference speed approximately five times faster than that of the transformer.

### 2.2 Context-Aware Sequential Recommendation Systems

SRS focuses on predicting the next item a user will interact with by analyzing the temporal sequence of items that the user has interacted with in the past (e.g., click, purchase, and cart). The initial sequential recommendation model used only the item-specific identifier sequence; therefore, the complex intentions and changing purchase environment of users were not reflected sufficiently [10,11]. With advances in deep learning models and computational hardware, processing unstructured data has become more efficient, and diverse data can be used. Consequently, recent research has explored models that can offer recommendations by integrating various types of context information, such as product information, user characteristics, and time background.

1. Studies incorporating product information typically use attributes such as category, brand, and text descriptions. FDSA [12] organizes category information into separate feature sequences, whereas S3-rec [13] uses various attributes such as location, keywords, and business type, as well as category information, as objects for self-supervised learning. Recently, graph neural network-based approaches have also been explored; Elahi et al. [14] proposed a graph attention-based collaborative filtering method that leverages knowledge graphs to capture item-specific characteristics.
2. Most studies that apply user characteristics use their demographic information (e.g., gender, age, and residence) to alleviate the “cold start” problem [15] by leveraging the behavioral patterns of groups with similar demographic characteristics for new users with insufficient data. For instance, Donkers et al. [16] proposed a model that includes user gender and age as input vectors, allowing more accurate identification of user preferences.
3. In SRS, time information is used to determine the order of item interactions and serves as additional information, such as the specific time when an interaction occurs or the time interval between items.

Because consumer preferences constantly change over time, temporal information plays a critical role in SRS: unlike product or user attributes, which remain largely fixed once observed, temporal signals can directly capture how a user's intent shifts across purchase occasions. Li et al. [17] proposed TiSASRec, a time-aware self-attention model, to sequentially learn the weights of various items and predict future items by integrating the absolute position and relative time interval between items.

Product and user characteristics are relatively static, whereas temporal information most directly captures changes in user preference over time. Therefore, integrating temporal information into SRS can improve personalized recommendation.

### 2.3 State Space Models

SSM represents time-series data using latent state variables. The basic system is defined by Eq. (1):

$$\begin{aligned}\dot{h}(t) &= Ah(t) + Bu(t), \\ y(t) &= Ch(t) + Du(t),\end{aligned}\tag{1}$$

where  $u(t)$  is the input signal,  $y(t)$  is the output signal, and  $h(t)$  represents the internal state of the system. The SSM model updates the internal state of the system through matrices  $A$ ,  $B$ ,  $C$ , and  $D$ . When a new signal is received, the system outputs a signal in accordance with the updated internal state. To apply this continuous-time formulation to sequential data, the system is discretized using a step size  $\Delta$  so that the internal state evolves recurrently over discrete time steps.

$$\begin{aligned}h_t &= \bar{A}h_{t-1} + \bar{B}x_t, \\ y_t &= Ch_t,\end{aligned}\tag{2}$$

where  $\bar{A}$  and  $\bar{B}$  are the discretized counterparts of  $A$  and  $B$ , respectively, and the term involving  $D$  is commonly omitted, as it merely acts as a direct skip connection from input to output.

Early SSMs faced difficulties in efficiently modeling long sequences because of the computational cost of high-dimensional state matrix  $A$  and optimization challenges. S4 [18] addressed long-range sequence modeling by initializing the state transition matrix using HiPPO [19]. S5 [20] subsequently improved the S4 architecture using multi-input, multi-output SSM and parallel scan. However, because the parameters  $\bar{A}$ ,  $\bar{B}$ , and  $C$  in Eq. (2) remain fixed regardless of the input, conventional SSM models are limited in their ability to capture input-dependent patterns. Mamba addresses this limitation through input-dependent selective state transitions and a hardware-aware selective-scan algorithm for efficient computation [21].

$$B_t = x_t W_B, C_t = x_t W_C, \Delta_t = \text{softplus}(x_t W_\Delta), \bar{A}_t = \exp(\Delta_t A).\tag{3}$$

While the efficiency of the selective scan mechanism was originally established in the general sequence modeling literature [21], recent studies have provided empirical evidence of this advantage specifically within the recommendation domain, alongside the emergence of hybrid and enhanced Mamba-based architectures for sequential recommendation. Liu et al. [22] proposed SIGMA, a bidirectional Mamba-based sequential recommender that addresses the unidirectional limitation of standard Mamba blocks, reporting consistent improvements over transformer- and Mamba-based baselines across five real-world datasets. Similarly, Zhang et al. [23] proposed a hyperbolic Mamba-based recommender that retains the linear-time efficiency of the Mamba architecture while achieving 3%–11% performance improvements across four benchmarks, demonstrating that recent extensions of the original Mamba block continue to deliver both efficiency and accuracy gains directly within the recommendation domain.

## 2.4 Research Gap and Proposed Approach

Despite these advances, existing time-aware models such as TiSASRec [17] primarily encode elapsed time and do not explicitly capture fixed-cycle periodicity, whereas SSM-based architectures such as S4 [18] and S5 [20] apply a fixed transition kernel regardless of the input, limiting their ability to adaptively weight historical time steps. Motivated by these gaps, this study proposes CalendarRec, which represents time using calendar-based categorical embeddings (month, day, day of week) rather than elapsed duration, incorporates the calendar context of the target prediction time, and conditions its Mamba-based selective scan on this temporal context to adaptively weight relevant historical interactions. Because the transition parameters  $B_t$ ,  $C_t$ , and  $\Delta_t$  in the selective scan (Eq. (3)) are computed directly from the input at each time step rather than fixed in advance, the model can adjust how strongly a given historical interaction is weighted based on the interaction itself—in our case, the combined item and calendar embedding. This input-dependent parameterization allows CalendarRec’s selective scan to modulate its memory update according to the calendar context of both past interactions and the target prediction time, rather than applying a uniform temporal decay or a fixed attention pattern regardless of when the interactions occurred.

## 3 Method

### 3.1 Dataset

This study used 2024 user click-log data for HFF products collected from GS SHOP [24], comprising 327,287 users, 8,257 products, and 2,305,071 clicks. The number of clicks per user ranged from 1 to 6,558 (Table 1).

**Table 1:** Statistics of the GS SHOP dataset (before preprocessing).

| Users   | Items | Interactions | Max. Length | Min. Length | Avg. Length |
|---------|-------|--------------|-------------|-------------|-------------|
| 327,287 | 8,257 | 2,305,071    | 6,558       | 1           | 7.04        |

Timestamps were recorded at one-second granularity. Click events with a stay time of  $\leq 1$  s were excluded, as such short durations are unlikely to reflect genuine user engagement and more likely result from accidental clicks or rapid scrolling. Each user’s entire remaining interaction history was treated as a single continuous sequence without explicit session boundaries. To train the model, the leave-one-out (LOO) method was applied to divide the data into training, validation, and test sets. The last product of each user was used as test data, the last product of the click sequence excluding the test data was used for validation, and the remaining data were used for training. To separate data using the LOO method, a sufficient sequence length is required. Accordingly, the minimum number of clicks was set to five, and users who clicked HFFs less than five times were excluded from the data. For stable learning, products that were clicked fewer than five times were also excluded [25,26]. The statistical characteristics of the dataset after preprocessing are summarized in Table 2.

**Table 2:** Statistics of the GS SHOP dataset (after preprocessing).

| Users   | Items | Interactions | Max. Length | Min. Length | Avg. Length |
|---------|-------|--------------|-------------|-------------|-------------|
| 113,335 | 6,285 | 1,902,616    | 6,558       | 5           | 16.79       |

### 3.2 Model Architecture

In this section, we explain the framework of the proposed model in detail. The proposed model comprises an Embedding layer, Mamba layer, and Embedding & prediction layer. First, the sequence of item click information and click times of the user are embedded into a latent vector space, and the sequence of

the item click history and times of the user are learned through an SSM-based Mamba block. The model operates by reflecting the time at which a recommendation list needs to be provided to the user and then ranking products according to their scores. The model framework is shown in Fig. 1.

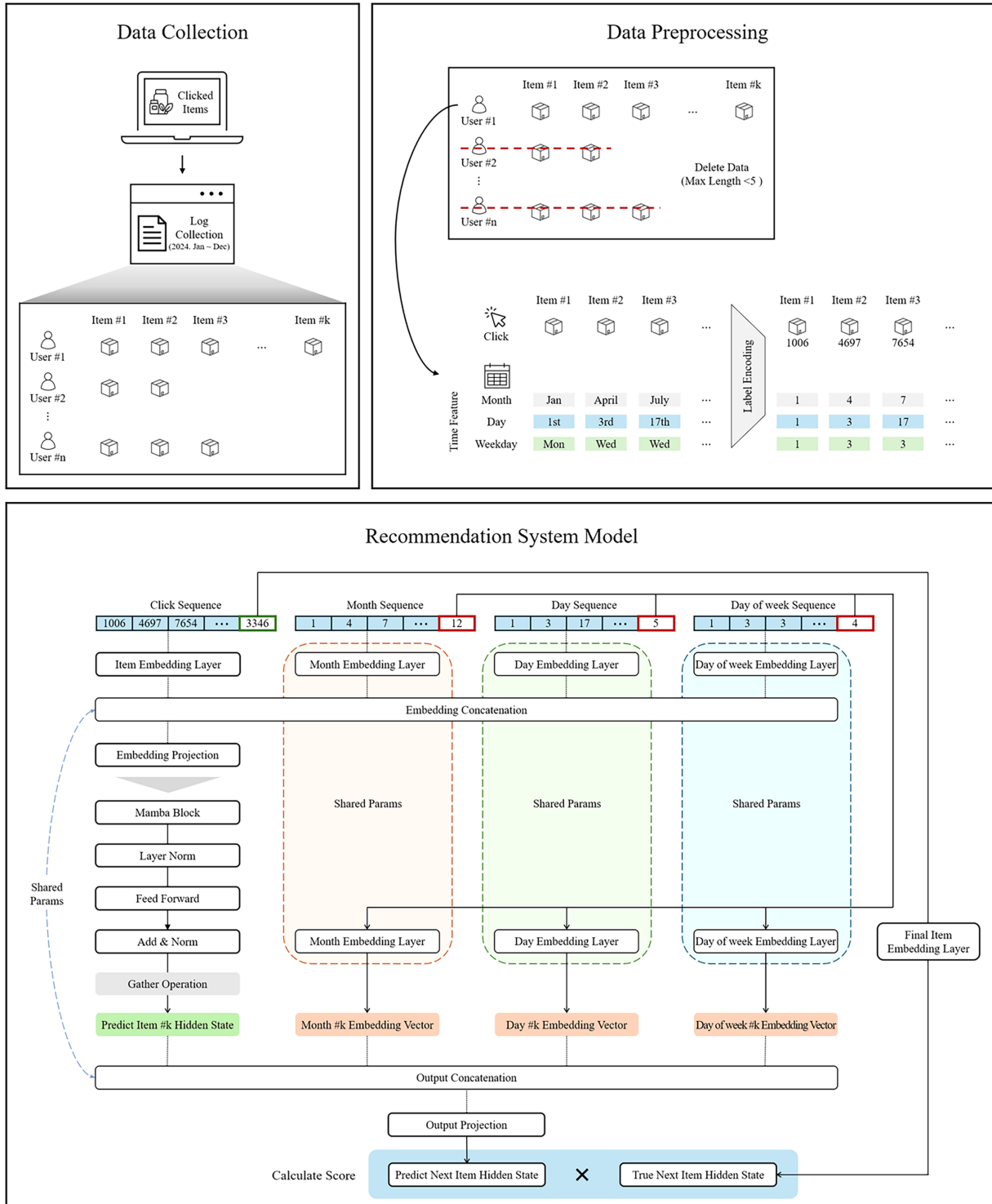


**Figure 1:** Model framework.

### 3.2.1 Embedding Layer

First, the sequence  $S_I$  of the item click history and each time sequence  $S_M, S_D$ , and  $S_W$  are embedded into independent latent spaces through the embedding layer.

$S_I$ : Sequence for item click information

$S_M$ : Sequence for the month that the item was clicked

$S_D$ : Sequence for the day that the item was clicked

$S_W$ : Sequence for the day of week that the item was clicked

The dimensions of each embedding vector in the latent space are given by  $E_I \in \mathbb{R}^{L \times H}$ ,  $E_M \in \mathbb{R}^{L \times H}$ ,  $E_D \in \mathbb{R}^{L \times H}$ ,  $E_W \in \mathbb{R}^{L \times H}$ , where  $L$  is the sequence length and  $H$  the dimensions of the embedding vector.

The embedding lookup tables for Month, Day, and Day of week are each learned independently of one another (i.e., they do not share parameters across the three calendar attributes). However, for each individual attribute, a single shared table is used to convert calendar values into embeddings regardless of whether the value belongs to the input sequence or the target prediction step. For instance, the same vector is used to represent “March” whether it appears in a past interaction or as the target prediction time. Periodicity is not modeled explicitly through a fixed functional form (e.g., sinusoidal encoding); instead, it is learned implicitly through the trainable categorical embeddings. No additional positional embedding is used, as the sequential recurrence of the Mamba block inherently encodes positional information.

Subsequently, each embedding vector is concatenated into a single embedding vector,  $E_C$ , and then reduced to the dimension of the original embedding vector,  $E_P$ , through a projection layer:

$$E_P = E_C W_P, E_C \in \mathbb{R}^{L \times 4H}, E_P \in \mathbb{R}^{L \times H}. \quad (4)$$

In addition, dropout and layer normalization are applied to increase learning stability and prevent overfitting. This prevents the embedding vectors from becoming excessively biased toward any single feature during learning.

### 3.2.2 Mamba Layer

The Mamba layer applies a selective SSM to the input  $E_P \in \mathbb{R}^{L \times H}$  through linear projection, 1D convolution, and Sigmoid Linear Unit (SiLU) activation. First,  $E_P$  is projected into two branches.

$$X = E_P W_X, \quad Z = E_P W_Z, \quad X, Z \in \mathbb{R}^{L \times H'} \quad (5)$$

$$X' = \text{SiLU}(\text{Conv1D}(X)). \quad (6)$$

The branch  $X$  is passed through a causal 1D convolution followed by a SiLU activation.

Subsequently, at each time step  $l$ , the SSM parameters  $B_l, C_l, \Delta_l$  are computed for  $X'$  and the output  $Y = \{y_1, \dots, y_L\}$  is obtained. In this process, SSM parameters are determined based on the input, which enables the model to effectively learn complex, long sequences. The SSM output is then combined with the gating branch  $Z$  and linearly projected back to the original dimension via  $W_{out}$

$$\hat{Y} = (Y \odot \text{SiLU}(Z)) W_{out}, \quad \hat{Y} \in \mathbb{R}^{L \times H}. \quad (7)$$

The resulting  $\hat{Y}$  is then passed through a residual connection and layer normalization to obtain the hidden representation  $H$  for the entire sequence.

$$H = \text{LayerNorm}(\text{Dropout}(\hat{Y}) + E_P). \quad (8)$$

Because the Mamba block has linear time complexity with respect to sequence length, inference efficiency comparable to that of RNNs is guaranteed, while a level of representation similar to that of transformer-based self-attention models is simultaneously secured through the selection mechanism. Subsequently, the positional feed forward network—which takes the output  $H$  of the Mamba block as input and consists of two fully connected layers with Gaussian Error Linear Unit (GELU) activation—additionally learns complex patterns through nonlinear transformations. Learning stability and generalization performance are secured through dropout and layer normalization, and this final output  $H^{(m)}$  constitutes the overall output of the Mamba layer.

$$\begin{aligned} F &= \text{GELU}(HW_1)W_2, \\ H^{(m)} &= \text{LayerNorm}(\text{Dropout}(F) + H). \end{aligned} \quad (9)$$

Here, the vector corresponding to each user's actual (i.e., non-padded) last interaction position is defined as  $H_t$ . That is,  $H_t$  is obtained by indexing  $H^{(m)}$  at the position corresponding to the user's most recent valid interaction and this vector is used for the next-item prediction described in [Section 3.2.3](#).

### 3.2.3 Embedding and Prediction Layer

In the embedding and prediction stages, the hidden state (i.e.,  $H_t$ ) corresponding to the final interaction point of the Mamba layer is used, along with the calendar context of the target prediction time. The target calendar values  $t_M$ ,  $t_D$  and  $t_W$  correspond to the date on which the recommendation would naturally be requested, that is, the date of the user's next visit to the platform, at which point the system would generate a recommendation list conditioned on the user's prior history and the current (i.e., that day's) calendar context. In the offline leave-one-out evaluation used in this study, this target calendar context is instantiated from the timestamp of the held-out interaction because this is the date on which the corresponding recommendation request would have occurred in a live deployment. Accordingly, the calendar context is supplied to the model as a known input reflecting the time of the request, and the model predicts only which item will be clicked—not when the interaction will occur.

These target calendar values are then embedded through an embedding layer into the same dimensional space as  $H_t$ .

$$e_{t_M}, e_{t_D}, e_{t_W} \in \mathbb{R}^{1 \times H}. \quad (10)$$

These time embeddings are then concatenated with  $H_t$  into a single vector  $E_{C_t}$ , and the vector is then reduced to the original embedding dimension via a projection layer. As in [Section 3.2.1](#), where the same embedding table is shared regardless of whether a calendar value belongs to the input sequence or the target step, the same projection layer  $W_P$  from the embedding stage ([Eq. \(4\)](#)) is reused here to reduce the concatenated  $4H$ -dimensional vector back to  $H$  dimensions, rather than introducing a separate set of parameters.

$$\begin{aligned} E_{C_t} &= \text{Concat}(H_t, e_{t_M}, e_{t_D}, e_{t_W}), \quad E_{C_t} \in \mathbb{R}^{1 \times 4H}, \\ E_{P_t} &= E_{C_t} W_P, \quad E_{P_t} \in \mathbb{R}^{1 \times H}. \end{aligned} \quad (11)$$

Subsequently, the scores for all candidate items are computed by taking the inner product between the resulting representation  $E_{P_t}$  and full item embedding matrix  $E_I^{\text{full}}$ . The score vector  $S$  in [Eq. \(12\)](#) is computed over the entire item catalog (6,285). That is, full-catalog ranking is used consistently during both training and evaluation phase.

$$S = E_{P_t} \left( E_I^{\text{full}} \right)^T, \quad S \in \mathbb{R}^{1 \times 6285}. \quad (12)$$

A probability distribution is then obtained using the softmax function, and the model is trained by minimizing the cross-entropy loss with respect to the ground-truth item  $I_t$  clicked at time step  $t$ .

$$p(I_t | H_t, t_M, t_D, t_W) = \frac{\exp(S_i)}{\sum_{j \in N} \exp(S_j)}, \quad \mathcal{L} = -\log p(I_t | H_t, t_M, t_D, t_W). \quad (13)$$

## 4 Experiments and Results

### 4.1 Experimental Environments

All experiments were conducted in a Linux environment using Python 3.10, PyTorch 2.8.0, and mamba-ssm 1.1.4. The hardware configuration consisted of an Intel Core i9-10980XE CPU and NVIDIA RTX 3090 GPU.

### 4.2 Hyperparameter Settings

To ensure a fair comparison, the proposed model and Mamba4Rec were configured with identical hyperparameters, as follows: a maximum sequence length of 10, 1 Mamba layer, and a hidden size of 64. The batch size was set to 1024, and the Adam optimizer was used with a learning rate of 0.001, and no weight decay was applied. The dropout rate was set to 0.2, and the cross-entropy loss function was employed. The model was trained for a maximum of 100 epochs, with early stopping applied if validation performance showed no improvement for 10 consecutive evaluations. For the remaining sequential recommendation baselines, each model's hyperparameter configuration as provided by RecBole [27] was used.

### 4.3 Random Seed Settings

For all experiments, a random seed of 2020 was used as the default seed for reporting main results, and nine additional seeds (2021–2029) were used to assess statistical significance across 10 independent runs.

### 4.4 Evaluation Metrics

#### 4.4.1 Hit Rate

Hit rate (HR) [28] measures whether a ground-truth item appears within the top-K recommendation list. It is intuitive but does not account for the rank of the item within the list.

#### 4.4.2 Normalized Discounted Cumulative Gain

Normalized discounted cumulative gain (NDCG) [29] evaluates recommendation quality while accounting for the rank of the ground-truth item; higher-ranked relevant items receive greater weight. It is defined as shown in Eq. (14) below.

$$\begin{aligned} DCG@K &= \sum_{i=1}^K \frac{2^{rel_i} - 1}{\log_2(i + 1)} \\ NDCG@K &= \frac{DCG@K}{IDCG@K}. \end{aligned} \quad (14)$$

#### 4.5 Comparison of Sequential Models

Following the seed configuration described in Section 4.3, Table 3 reports representative results obtained using the base seed (2020) for all models, whereas Table 4 reports the mean  $\pm$  standard deviation over the 10 independent runs for Mamba4Rec and the proposed model, together with the corresponding paired  $t$ -test results. The proposed model was compared with GRU4Rec, SASRec, BERT4Rec, Mamba4Rec, and TiSASRec, with performance and efficiency results summarized in Table 3 and comparisons across sequence lengths visualized in Fig. 2. For a fair comparison, the same Mamba configuration (1 layer and a hidden size of 64) was used for the proposed model and Mamba4Rec [9].

**Table 3:** Performance comparison of sequential recommendation models.

|             |            | Performance   |               |               |               |               | Lightweight |           |                 |                        |
|-------------|------------|---------------|---------------|---------------|---------------|---------------|-------------|-----------|-----------------|------------------------|
|             |            | HR@1          | HR@5          | HR@10         | NDCG@5        | NDCG@10       | # of Param  | MFLOPs    | GPU Memory (MB) | Inference Latency (ms) |
| Length = 10 | GRU4Rec    | 0.355         | 0.5904        | 0.666         | 0.482         | 0.5065        | 484,224     | 828,864   | 348.3687        | 0.891                  |
|             | SASRec     | 0.3556        | 0.5645        | 0.6368        | 0.4687        | 0.4921        | 502,976     | 991,944   | 329.0254        | 1.827                  |
|             | BERT4Rec   | 0.3565        | 0.5724        | 0.642         | 0.4738        | 0.4962        | 513,613     | 1,034,184 | 329.0259        | 3.224                  |
|             | TiSASRec   | 0.357         | 0.5672        | 0.6447        | 0.4703        | 0.4955        | 536,512     | 1,005,384 | 287.2085        | 1.730                  |
|             | Mamba4Rec  | 0.3593        | 0.5885        | 0.6601        | 0.4836        | 0.5068        | 474,496     | 332,224   | 329.0171        | 0.729                  |
|             | Our Method | <u>0.4327</u> | <u>0.6621</u> | <u>0.723</u>  | <u>0.5577</u> | <u>0.5774</u> | 514,944     | 514,560   | 329.0366        | 0.891                  |
| Length = 30 | GRU4Rec    | 0.3532        | 0.5878        | 0.6629        | 0.4799        | 0.5043        | 484,224     | 2,486,464 | 348.5396        | 0.966                  |
|             | SASRec     | 0.3556        | 0.5666        | 0.6408        | 0.4693        | 0.4932        | 504,256     | 2,982,904 | 331.085         | 2.03                   |
|             | BERT4Rec   | 0.3569        | 0.5726        | 0.6442        | 0.474         | 0.4972        | 514,893     | 3,109,624 | 331.082         | 3.63                   |
|             | TiSASRec   | 0.3559        | 0.5695        | 0.6434        | 0.4711        | 0.4951        | 539,072     | 3,100,024 | 287.6899        | 1.76                   |
|             | Mamba4Rec  | 0.36          | 0.5876        | 0.6594        | 0.4836        | 0.5069        | 474,496     | 996,544   | 331.0513        | 0.74                   |
|             | Our Method | <u>0.4327</u> | <u>0.6622</u> | <u>0.725</u>  | <u>0.5577</u> | <u>0.5782</u> | 514,944     | 1,510,400 | 331.1099        | 0.921                  |
| Length = 50 | GRU4Rec    | 0.3535        | 0.59          | 0.6651        | 0.4809        | 0.5053        | 484,224     | 4,144,064 | 348.7104        | 1.245                  |
|             | SASRec     | 0.3549        | 0.5654        | 0.6392        | 0.4687        | 0.4926        | 505,536     | 4,983,464 | 331.1592        | 2.545                  |
|             | BERT4Rec   | 0.3548        | 0.5704        | 0.6407        | 0.4721        | 0.4949        | 516,173     | 5,194,664 | 331.1504        | 4.022                  |
|             | TiSASRec   | 0.3551        | 0.5686        | 0.6431        | 0.4704        | 0.4945        | 541,632     | 5,306,664 | 288.5903        | 2.12                   |
|             | Mamba4Rec  | 0.3582        | 0.589         | 0.6628        | 0.4838        | 0.5078        | 474,496     | 1,660,864 | 331.0854        | 0.783                  |
|             | Our Method | <b>0.4351</b> | <b>0.663</b>  | <b>0.7258</b> | <b>0.5593</b> | <b>0.5798</b> | 514,944     | 2,506,240 | 331.1831        | 0.95                   |

Note: The **bold** values indicate the best performance across all Max Length settings, and the underlined values indicate the best performance within each Max Length setting.

**Table 4:** Performance and statistical significance of Mamba4Rec and the proposed model across sequence lengths.

| Length | Model      | HR@1                | HR@10               | NDCG@10             | vs. Mamba4Rec                              |
|--------|------------|---------------------|---------------------|---------------------|--------------------------------------------|
| 10     | Mamba4Rec  | 0.3578 $\pm$ 0.0011 | 0.6582 $\pm$ 0.002  | 0.5052 $\pm$ 0.0011 | HR@1: $p < 0.001$                          |
|        | Our Method | 0.4329 $\pm$ 0.0007 | 0.7238 $\pm$ 0.0008 | 0.5780 $\pm$ 0.0005 | HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ |
| 30     | Mamba4Rec  | 0.3577 $\pm$ 0.0013 | 0.6601 $\pm$ 0.002  | 0.5060 $\pm$ 0.0012 | HR@1: $p < 0.001$                          |
|        | Our Method | 0.4334 $\pm$ 0.0014 | 0.7255 $\pm$ 0.0008 | 0.5790 $\pm$ 0.0007 | HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ |
| 50     | Mamba4Rec  | 0.3579 $\pm$ 0.0009 | 0.6615 $\pm$ 0.0015 | 0.5068 $\pm$ 0.0009 | HR@1: $p < 0.001$                          |
|        | Our Method | 0.4330 $\pm$ 0.0014 | 0.7253 $\pm$ 0.001  | 0.5787 $\pm$ 0.0006 | HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ |

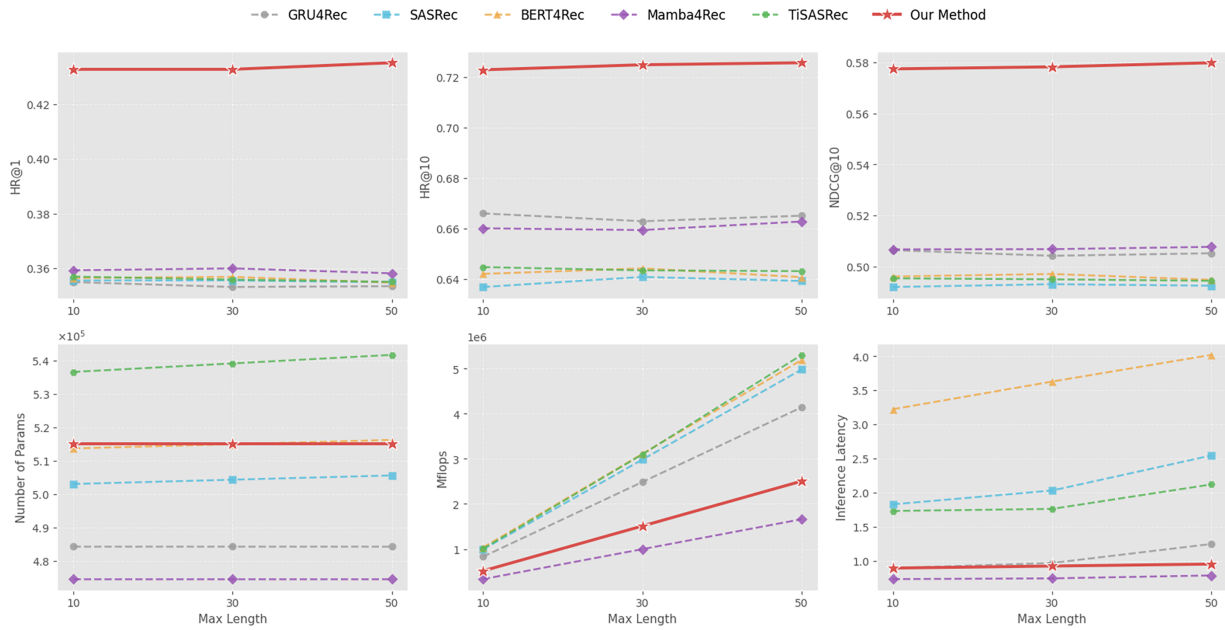


**Figure 2:** Effectiveness (HR@1, HR@10, NDCG@10) and efficiency (number of parameters, MFLOPs, inference latency) of sequential recommendation models under varying maximum sequence lengths.

GRU4Rec showed strong performance in the top 5 and 10 recommendation results, outperforming models that use the transformer structure regardless of the click sequence length, with HR@10 of 0.666 and NDCG@10 of 0.5065 at a click sequence length of 10. This is likely because the dataset's relatively short average sequence length favored the inductive bias of recurrent architectures over self-attention, which typically requires longer sequences and larger-scale data to be fully effective. Meanwhile, BERT4Rec outperformed SASRec in most performance metrics on the GS SHOP dataset, indicating that learning bidirectionally to analyze the overall context is more effective than learning unidirectionally. TiSASRec, which extends SASRec by incorporating relative time-interval information between interactions, achieved performance comparable to BERT4Rec across most metrics and sequence lengths (e.g., NDCG@10 of 0.4955 at length 10) but did not clearly surpass it. This result suggests that modeling time intervals alone is not sufficient on its own.

Mamba4Rec, a widely used SSM, outperformed GRU4Rec in terms of NDCG, and transformer-based models across most metrics. It recorded an NDCG@10 of 0.5078 with a max length of 50, showing relative improvements of 0.5% and 2.6% over GRU4Rec and BERT4Rec, respectively. The proposed model demonstrated the best performance across all metrics, regardless of the click sequence length. At a click sequence length of 50, it recorded 0.4351, 0.663, and 0.7258 in terms of HR@1, HR@5, and HR@10, respectively. It achieved the highest performance of 0.5593 and 0.5798 in terms of NDCG@5 and NDCG@10, respectively, representing a relative improvement of 9.5% in HR@10 and 14.2% in NDCG@10 over Mamba4Rec, which does not consider time information. This result demonstrates that incorporating time information is associated with improved recommendation quality under the tested conditions when recommending product information to users. To further validate the improvement over the base architecture, we conducted paired *t*-tests between Mamba4Rec and the proposed model over 10 random seeds (2020–2029) at each sequence length (Table 4). The proposed model achieved statistically significant improvements over Mamba4Rec across all three lengths and all three metrics (all  $p < 0.001$ ), confirming that the observed performance gains are highly robust and not attributable to random variation in model initialization or training.

In terms of model efficiency, Mamba4Rec had the fewest parameters and lowest computational cost. The proposed model incurred a modest increase in computational cost relative to Mamba4Rec (approximately +8.5% in parameter count and +50.9% in MFLOPs at max sequence length 50, due to the additional temporal embedding layers) but remained substantially lighter than the transformer-based models (SASRec and BERT4Rec), whose MFLOPs were approximately  $2\times$  higher than those of the proposed model. Thus, although not as lightweight as its base architecture, the proposed model retains a favorable efficiency profile relative to the transformer-based alternatives. Table 3 further reports GPU memory usage and inference latency measured under a batch size of 1, reflecting a real-time serving scenario. Across all sequence lengths, the proposed model achieved inference latency approximately  $1.9\text{--}4.2\times$  lower than the transformer-based baselines (SASRec, BERT4Rec, and TiSASRec); at a sequence length of 50, for instance, the proposed model recorded a latency of 0.95 ms, compared to 2.12–4.02 ms for these models. Relative to Mamba4Rec, the proposed model exhibited a modest latency increase of approximately 21.3% (0.95 vs. 0.783 ms), consistent with the parameter and MFLOPs comparisons discussed above. GPU memory usage in Table 3 denotes the total process-level GPU memory footprint required to serve a single inference request (batch size = 1) after the model has been loaded, computed as the sum of (i) the process's baseline GPU memory immediately after model loading, which includes model parameters and CUDA context initialization overhead, and (ii) the additional peak activation memory allocated during the subsequent inference call. Under this measurement, GPU memory usage ranged from approximately 287 to 349 MB across all models, with no substantial differences among most models. GRU4Rec showed a somewhat higher value, likely reflecting the sequential nature of its recurrent computation.

The proposed model, which integrates time information, goes beyond simple sequential click log records of “which item was clicked” to reflect the temporal context of “when it was clicked,” enabling more accurate learning of user behavior patterns. Overall, these results indicate that the modest increase in model size and computation is well justified by the resulting performance gains.

## 4.6 Ablation Study

### 4.6.1 Comparison of Time Feature Combinations

All ablation experiments used the hyperparameter settings described in Section 4.2. The effects of Day of week, Day, and Month features were evaluated individually and in combination, with results presented in Table 5 and Fig. 3.

**Table 5:** Performance comparison of time feature combinations.

| Order   | 1st         |        |               | 2nd                  |                        |                | 3rd                          |
|---------|-------------|--------|---------------|----------------------|------------------------|----------------|------------------------------|
| Feature | Day of Week | Day    | Month         | Day of Week<br>+ Day | Day of Week<br>+ Month | Day +<br>Month | Day of Week +<br>Day + Month |
| HR@1    | 0.3662      | 0.3733 | <u>0.4004</u> | 0.4245               | 0.4098                 | <u>0.4314</u>  | <b>0.4327</b>                |
| HR@5    | 0.6191      | 0.6138 | <u>0.6379</u> | 0.6555               | 0.6557                 | <u>0.6611</u>  | <b>0.6621</b>                |
| HR@10   | 0.6906      | 0.6842 | <u>0.7017</u> | 0.7149               | 0.7163                 | <u>0.7225</u>  | <b>0.723</b>                 |
| NDCG@5  | 0.5031      | 0.5035 | <u>0.5294</u> | 0.5508               | 0.5446                 | <u>0.5566</u>  | <b>0.5577</b>                |
| NDCG@10 | 0.5264      | 0.5264 | <u>0.5501</u> | 0.5701               | 0.5643                 | <u>0.5766</u>  | <b>0.5774</b>                |

Note: The **bold** values indicate the best performance across all feature combinations, and the underlined values indicate the best performance within each order of feature combinations, where “order” refers to the number of calendar features combined (e.g., “first-order” denotes a single feature, “second-order” denotes two features combined) rather than a formal statistical notion of feature-interaction order.

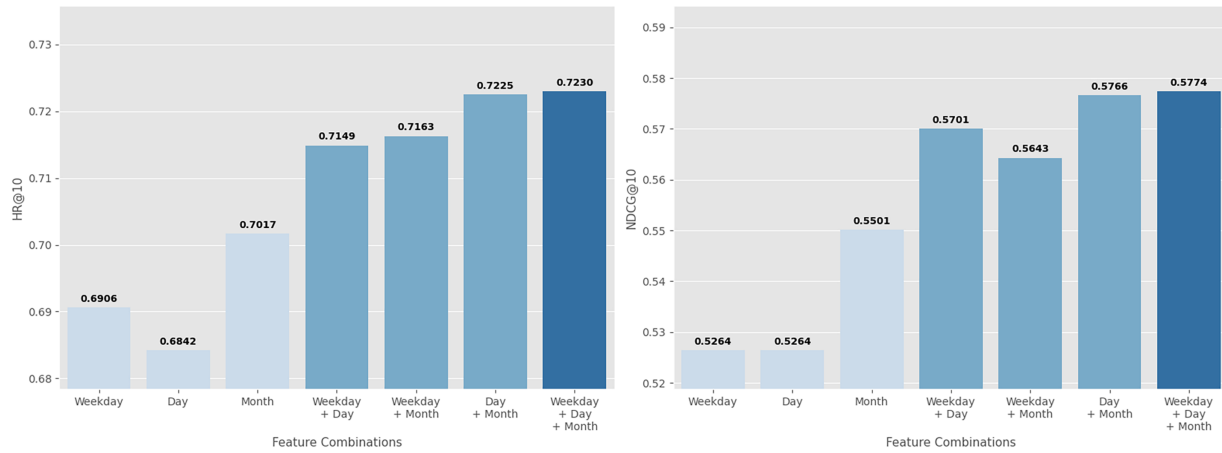


**Figure 3:** Comparison of time feature combinations in terms of HR@10 and NDCG@10.

The single time characteristic was applied first, and the highest performance was observed with the Month variable. NDCG@10 showed a relative increase of 4.5% compared with both the Day of week and Day variables. Next, we combined second-order time characteristics. The combination of Day and Month showed the best performance, followed by Day of week + Day and Day of week + Month in terms of NDCG, although the ranking between these two combinations varied slightly across HR-based metrics. The NDCG@10 results showed a relative increase of approximately 1.1% over the combination of Day of week and Day, and approximately 2.2% over the combination of Day of week and Month. This pattern may be related to the fixed-period packaging of HFFs (e.g., 30- or 60-day supplies), although this interpretation was not directly tested through repurchase-timing analysis and is instead grounded in the fixed-cycle sales convention of HFF products. Finally, the best performance was observed when all three temporal characteristics (Day of week, Day, and Month) were combined. The performance showed a relative improvement of approximately 5% over the best performing single feature (Month) in terms of NDCG@10.

Table 6 reports pairwise paired *t*-tests among the three temporal feature configurations (Month alone, Day + Month, and the full combination) over 10 random seeds. Compared with Month alone, both the Day + Month combination and the full combination yielded statistically significant improvements across all metrics (all  $p < 0.001$ ). However, when comparing the Day + Month combination directly against the full combination, adding the Day of week feature produced a clear additional improvement in HR@10 ( $p = 0.008$ ) and NDCG@10 ( $p = 0.009$ ), while the corresponding improvement in HR@1 did not reach significance ( $p = 0.091$ ). Therefore, adding the Day of week feature had limited effect on top 1 prediction accuracy but was effective in improving the quality of the top 10 recommendation list.

**Table 6:** Performance and pairwise statistical significance of temporal feature combinations.

| Feature                     | HR@1            | HR@10           | NDCG@10         | vs. 1st Order                                                   | vs. 2nd Order                                                   |
|-----------------------------|-----------------|-----------------|-----------------|-----------------------------------------------------------------|-----------------------------------------------------------------|
| <b>1st: Month</b>           | 0.4005 ± 0.0007 | 0.7009 ± 0.0007 | 0.5497 ± 0.0004 | –                                                               | –                                                               |
| <b>2nd: Day + Month</b>     | 0.4320 ± 0.0012 | 0.7226 ± 0.0007 | 0.5771 ± 0.0006 | HR@1: $p < 0.001$<br>HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ | –                                                               |
| <b>3rd: Full (Proposed)</b> | 0.4329 ± 0.0007 | 0.7238 ± 0.0008 | 0.5780 ± 0.0005 | HR@1: $p < 0.001$<br>HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ | HR@1: $p = 0.091$<br>HR@10: $p = 0.008$<br>NDCG@10: $p = 0.009$ |

#### 4.6.2 Comparison of Hyperparameters

The effects of Mamba depth, sequence length, and embedding dimension were evaluated using 1–3 layers; sequence lengths of 10, 30, and 50; and embedding dimensions of 64, 128, and 256. Results are presented in Table 7 and Fig. 4.

**Table 7:** Comparison of hyperparameter settings.

| # of Layers |         | 1      |        |        | 2             |               |        | 3             |               |        |
|-------------|---------|--------|--------|--------|---------------|---------------|--------|---------------|---------------|--------|
| Hidden Size |         | 64     | 128    | 256    | 64            | 128           | 256    | 64            | 128           | 256    |
| Length = 10 | HR@1    | 0.4327 | 0.4307 | 0.4248 | <u>0.4395</u> | 0.4366        | 0.433  | 0.4382        | 0.4365        | 0.4331 |
|             | HR@5    | 0.6621 | 0.6635 | 0.6632 | 0.6682        | <u>0.6695</u> | 0.6667 | 0.6691        | 0.6683        | 0.6674 |
|             | HR@10   | 0.723  | 0.7258 | 0.7262 | 0.7309        | <u>0.7318</u> | 0.7298 | <u>0.7319</u> | 0.7308        | 0.7305 |
|             | NDCG@5  | 0.5577 | 0.5579 | 0.555  | <u>0.5643</u> | 0.5636        | 0.5605 | 0.564         | 0.563         | 0.561  |
|             | NDCG@10 | 0.5774 | 0.5782 | 0.5755 | <u>0.5847</u> | 0.5839        | 0.581  | 0.5845        | 0.5833        | 0.5815 |
| Length = 30 | HR@1    | 0.4327 | 0.4305 | 0.4289 | <u>0.4405</u> | 0.4376        | 0.4354 | 0.4394        | 0.4369        | 0.4299 |
|             | HR@5    | 0.6622 | 0.6644 | 0.663  | 0.6696        | <u>0.6699</u> | 0.6681 | <u>0.6699</u> | <u>0.6699</u> | 0.66   |
|             | HR@10   | 0.725  | 0.7267 | 0.7247 | 0.7321        | <u>0.7335</u> | 0.7301 | 0.7325        | 0.7321        | 0.7187 |
|             | NDCG@5  | 0.5577 | 0.5582 | 0.5567 | <u>0.5655</u> | 0.5644        | 0.5625 | 0.5652        | 0.5639        | 0.5558 |
|             | NDCG@10 | 0.5782 | 0.5784 | 0.5768 | <u>0.5858</u> | 0.5851        | 0.5827 | 0.5855        | 0.5841        | 0.5748 |
| Length = 50 | HR@1    | 0.4351 | 0.4299 | 0.4298 | 0.4394        | 0.4367        | 0.4326 | <u>0.44</u>   | 0.4361        | 0.4375 |
|             | HR@5    | 0.663  | 0.6642 | 0.6634 | 0.6689        | <u>0.6709</u> | 0.6674 | 0.6705        | 0.6706        | 0.6685 |
|             | HR@10   | 0.7258 | 0.7265 | 0.7262 | 0.7322        | <u>0.7345</u> | 0.7308 | 0.7325        | 0.7334        | 0.7316 |
|             | NDCG@5  | 0.5593 | 0.5579 | 0.5572 | 0.5647        | 0.5644        | 0.5606 | <u>0.5657</u> | 0.564         | 0.5634 |
|             | NDCG@10 | 0.5798 | 0.5782 | 0.5777 | 0.5853        | 0.5851        | 0.5812 | <u>0.5858</u> | 0.5844        | 0.5839 |

Note: The underlined values indicate the best performance within each length.

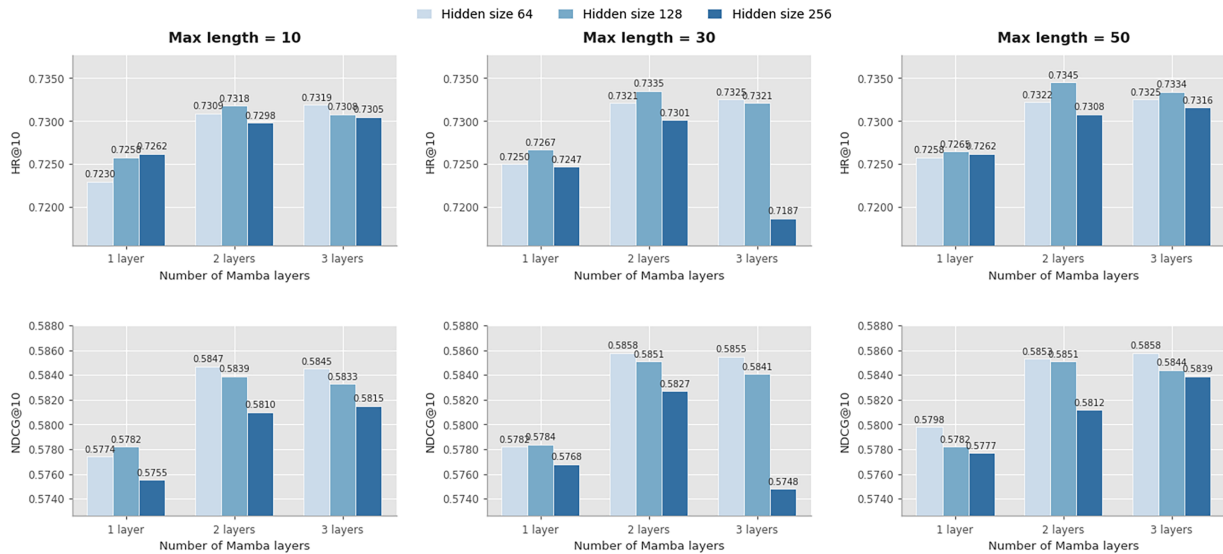


**Figure 4:** Comparison of hyperparameter settings in terms of HR@10 and NDCG@10.

**Table 8:** Performance and statistical significance of hyperparameter configurations.

| Length | # of Layers | HR@1                                  | HR@10                                 | NDCG@10                               | vs. Best Combination                                            |
|--------|-------------|---------------------------------------|---------------------------------------|---------------------------------------|-----------------------------------------------------------------|
| 10     | 1           | $0.4329 \pm 0.0007$                   | $0.7238 \pm 0.0008$                   | $0.5780 \pm 0.0005$                   | HR@1: $p < 0.001$<br>HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ |
|        | 2           | $0.4396 \pm 0.0011$                   | $0.7310 \pm 0.0006$                   | $0.5849 \pm 0.0004$                   | HR@1: $p = 0.236$<br>HR@10: $p < 0.001$<br>NDCG@10: $p = 0.003$ |
|        | 3           | $0.4386 \pm 0.0007$                   | $0.7316 \pm 0.0005$                   | $0.5847 \pm 0.0003$                   | HR@1: $p < 0.001$<br>HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ |
| 30     | 1           | $0.4334 \pm 0.0014$                   | $0.7255 \pm 0.0008$                   | $0.5790 \pm 0.0007$                   | HR@1: $p < 0.001$<br>HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ |
|        | 2           | $0.4395 \pm 0.0010$                   | $0.7325 \pm 0.0007$                   | $0.5855 \pm 0.0006$                   | HR@1: $p = 0.199$<br>HR@10: $p = 0.026$<br>NDCG@10: $p = 0.372$ |
|        | 3           | $0.4390 \pm 0.0006$                   | $0.7330 \pm 0.0008$                   | $0.5854 \pm 0.0004$                   | HR@1: $p = 0.005$<br>HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ |
| 50     | 1           | $0.4330 \pm 0.0014$                   | $0.7253 \pm 0.0010$                   | $0.5787 \pm 0.0006$                   | HR@1: $p < 0.001$<br>HR@10: $p < 0.001$<br>NDCG@10: $p < 0.001$ |
|        | 2           | <u><math>0.4402 \pm 0.0007</math></u> | $0.7322 \pm 0.0004$                   | $0.5856 \pm 0.0003$                   | HR@1: –<br>HR@10: $p = 0.005$<br>NDCG@10: $p = 0.684$           |
|        | 3           | $0.4392 \pm 0.0009$                   | <u><math>0.7333 \pm 0.0006</math></u> | <u><math>0.5857 \pm 0.0005</math></u> | HR@1: $p = 0.058$<br>HR@10: –<br>NDCG@10: –                     |

Note: The underlined values indicate the best performance across all parameter combinations.

Experimental results showed that for a sequence length of 50, the best results were observed across all NDCG@K when the number of Mamba layers was 3, and the embedding dimension was 64. For a sequence length of 30, the best performance was observed across all metrics when the number of Mamba layers was 2. For a sequence length of 10, the best HR@1, NDCG@5, and NDCG@10 performance was achieved with 2 Mamba layers, while HR@10 was marginally higher with 3 layers, a difference of only 0.0001, which is unlikely to be practically meaningful; taken together, these results indicate that 2 layers remains the more consistent choice for shorter sequences.

When the embedding dimension was 64, results across all NDCG@K were the best among all combinations. By contrast, when the number of Mamba layers was 1 or the embedding dimension was 256, the performance was the lowest compared to other combinations of hyperparameters. This trend suggests that

while a deep model is necessary to learn user behavior patterns, performance may suffer owing to issues such as overfitting if the embedding dimension increases. This ablation study confirmed that a relatively shallow network (2-layer, 64-dimension) is suitable for short sequences, whereas a deep network (3-layer, 64-dimension) is effective for long sequences; thus, a balance between model complexity and sequence length is important for recommendation performance.

With hidden size fixed at 64, we evaluated all nine combinations of sequence length (10, 30, 50) and number of Mamba layers (1, 2, 3) over 10 random seeds (Table 8). The best-performing configuration differed slightly by metric: length = 50/layers = 2 achieved the highest HR@1, whereas length = 50/layers = 3 achieved the highest HR@10 and NDCG@10. Using each metric's own best-performing configuration as the reference, paired *t*-tests (Bonferroni-corrected for multiple comparisons) revealed a more nuanced pattern than a simple length- or depth-based rule: configurations with 1 layer were consistently and significantly outperformed by the reference configuration across all sequence lengths and metrics. Among 2- and 3-layer configurations, results diverged by both sequence length and metric. At length = 30, the 2-layer configuration showed no significant difference from the reference on any of the three metrics, whereas the 3-layer configuration at the same length differed significantly on all three metrics (HR@1:  $p = 0.005$ ; HR@10:  $p < 0.001$ ; NDCG@10:  $p < 0.001$ ) despite its comparable raw performance. At length = 50, the pattern was metric-dependent rather than uniformly nonsignificant: the 2-layer configuration was significantly different from the reference on HR@10 ( $p = 0.005$ ) but not on NDCG@10 ( $p = 0.684$ ), whereas the 3-layer configuration approached but did not reach significance on HR@1 ( $p = 0.058$ ). These results indicate that increasing the number of Mamba layers or the input sequence length beyond a certain point yield diminishing returns in performance. However, it would be an overgeneralization to conclude that "all high-performing configurations are statistically indistinguishable." In practice, results diverged depending on the specific combination of sequence length, layer depth, and evaluation metric: for some configurations (e.g., length = 30/layers = 2), the apparent performance difference fell within the range of run-to-run variance, whereas for others (e.g., length = 30/layers = 3), configurations with comparable raw performance still showed statistically significant differences from the reference.

#### 4.6.3 Comparison of Parameter Sharing Strategies

As described in Sections 3.2.1 and 3.2.3, our model shares parameters at two points: the calendar embedding tables across the input sequence and target step, and the projection layer across the embedding and prediction stages. This subsection examines the individual contribution of each sharing strategy.

Table 9 presents the results of the ablation study on parameter sharing across 10 random seeds. Removing either the shared time embedding tables or the shared projection layer led to statistically significant performance degradation across all three metrics (all  $p < 0.05$ ), confirming that both sharing strategies contribute meaningfully and independently to the overall performance of the proposed model. Moreover, the proposed model achieved superior performance across all three metrics despite having fewer trainable parameters than either non-shared variant (514,944 vs. 534,784 and 531,392). Therefore, the performance gain does not stem from increased model capacity, but rather from the shared parameters enforcing a consistent representation across different stages of the model.

**Table 9:** Performance and statistical significance of parameter sharing configurations.

|                              | HR@1                | HR@10               | NDCG@10             | # of Params | vs. Proposed Model                                              |
|------------------------------|---------------------|---------------------|---------------------|-------------|-----------------------------------------------------------------|
| Proposed Model               | 0.4329 $\pm$ 0.0007 | 0.7238 $\pm$ 0.0008 | 0.5780 $\pm$ 0.0005 | 514,944     | –                                                               |
| w/o Time Embedding Sharing   | 0.4314 $\pm$ 0.0011 | 0.7228 $\pm$ 0.0007 | 0.5769 $\pm$ 0.0008 | 534,784     | HR@1: $p = 0.008$<br>HR@10: $p = 0.006$<br>NDCG@10: $p = 0.002$ |
| w/o Projection Layer Sharing | 0.4313 $\pm$ 0.0012 | 0.7227 $\pm$ 0.0005 | 0.5766 $\pm$ 0.0006 | 531,392     | HR@1: $p = 0.006$<br>HR@10: $p = 0.003$<br>NDCG@10: $p < 0.001$ |

Note: All other hyperparameter settings follow those described in [Section 4.2](#).

#### 4.7 Generalization to a Public Dataset

To address the limitations that the preceding experiments were confined to domain-specific datasets and verify the generalizability of the proposed method, additional experiments were conducted on widely used public datasets, namely the Amazon review datasets (Beauty, Video Games, CD, and Movie). GRU4Rec, SASRec, BERT4Rec, TiSASRec, and Mamba4Rec were used as baselines, with HR@10 and NDCG@10 as evaluation metrics. The maximum sequence length was set to 50 for all datasets, and proposed model's hyperparameters were selected by grid search. The experimental results are presented in [Table 10](#).

**Table 10:** Performance comparison for generalization to public datasets.

|            | Amazon-Beauty |               | Amazon-Video-Games |               | Amazon-Cd     |               | Amazon-Movie  |               |
|------------|---------------|---------------|--------------------|---------------|---------------|---------------|---------------|---------------|
|            | HR@10         | NDCG@10       | HR@10              | NDCG@10       | HR@10         | NDCG@10       | HR@10         | NDCG@10       |
| GRU4Rec    | 0.0602        | 0.033         | 0.1027             | 0.0535        | 0.0681        | 0.0368        | 0.0867        | 0.0507        |
| SASRec     | 0.0838        | 0.0412        | 0.1135             | 0.0549        | 0.0943        | 0.043         | 0.0979        | 0.0502        |
| BERT4Rec   | 0.0379        | 0.0188        | 0.0795             | 0.04          | 0.053         | 0.0266        | 0.0711        | 0.0369        |
| TiSASRec   | <u>0.0841</u> | <u>0.0423</u> | 0.1176             | 0.0557        | 0.0958        | 0.0443        | <u>0.0981</u> | 0.0513        |
| Mamba4Rec  | 0.0661        | 0.0416        | 0.1034             | 0.0571        | 0.0783        | 0.0431        | 0.092         | 0.0542        |
| Our Method | 0.067         | 0.0415        | <u>0.1267</u>      | <u>0.0715</u> | <u>0.0979</u> | <u>0.0565</u> | 0.0966        | <u>0.0585</u> |

Note: The underlined values indicate the best performance within each dataset.

Among the four public datasets, the proposed method (Our Method) achieved the best performance in terms of HR@10 and NDCG@10 on the Amazon-Video-Games and Amazon-Cd datasets. In particular, on the Amazon-Video-Games dataset, the proposed method recorded an HR@10 of 0.1267 and an NDCG@10 of 0.0715, corresponding to relative improvements of approximately 7.7% and 28.4% over the second-best model, TiSASRec. On the Amazon-Cd dataset, the proposed method likewise achieved solid improvements with an HR@10 of 0.0979 and an NDCG@10 of 0.0565, compared to TiSASRec's HR@10 of 0.0958 and NDCG@10 of 0.0443.

On the Amazon-Beauty dataset, TiSASRec (HR@10 of 0.0841, NDCG@10 of 0.0423) achieved the best performance. However, the proposed method (HR@10 of 0.067, NDCG@10 of 0.0415) still delivered competitive results that outperformed other baselines such as GRU4Rec, BERT4Rec, and Mamba4Rec. On the Amazon-Movie dataset, TiSASRec slightly outperformed the proposed method in terms of HR@10 (0.0981), but the proposed method achieved the highest NDCG@10 (0.0585), demonstrating superior ranking quality among the top recommended items.

Overall, the proposed method achieved the best performance in terms of both metrics on two of the four public datasets, while remaining highly competitive on the other two. These results provide additional supporting evidence that the proposed method is not overfitted to a specific domain and generalizes reasonably well across diverse domains and data characteristics; however, performance was not uniformly superior across all four categories.

## 5 Conclusion and Future Work

We proposed CalendarRec, a sequential recommendation model that incorporates multidimensional temporal context based on the Mamba architecture, and validated its effectiveness using real-world click log data from an e-commerce platform. Experimental results showed that integrating temporal context enabled CalendarRec to achieve relative gains of 9.5% in HR@10 and 14.2% in NDCG@10 over Mamba4Rec. Our ablation results suggest that, although the underlying mechanism warrants further investigation, temporal context meaningfully contributes to recommendation performance in this dataset. Additional analysis further revealed that the optimal Mamba depth depends on sequence length, indicating that increasing model complexity does not necessarily yield better performance.

The primary academic and practical implications derived from this study are as follows. First, this study successfully integrated the Mamba architecture into the e-commerce recommendation landscape. It addresses the computational efficiency issues of conventional models and expands the research scope of context-aware recommendation systems by proposing a specific methodology for modeling multidimensional temporal context. CalendarRec captures the interaction between long-term cycles and short-term routines by integrating multi-order temporal features. Second, CalendarRec demonstrated high predictive accuracy with substantially lower computational overhead than transformer-based models, although modestly higher than that of its Mamba4Rec baseline. Therefore, it is a practical model optimized for real-world e-commerce environments where real-time responsiveness is critical. By enabling timely recommendations in the HFF domain—where cyclical interaction patterns are prominent—it serves as a practical tool for enhancing marketing efficiency and customer retention. Third, the proposed architecture accommodates several practical deployment settings that may arise in real-world e-commerce environments, though a full empirical evaluation of these settings is beyond the scope of this study and was therefore not separately conducted. In particular, the model relies solely on behavioral information rather than user identity, which allows it to support anonymous users. This flexibility extends to the input sequence itself, which can vary in length across users; because the Mamba layer's hidden state is updated incrementally, new interactions can also be incorporated as they occur without reprocessing the entire sequence. Together, these properties make the model well suited to serving recommendations continuously as user behavior evolves. In deployment, the target calendar context would similarly be instantiated from the date of the incoming recommendation request, so that recommendations remain aligned with the day on which they are served; in the offline evaluation reported here, this context is instead instantiated from the timestamp of the held-out interaction, which serves as the simulated request time (Section 3.2.3). Fourth, the proposed time-based recommendation mechanism is hypothesized to generalize to other domains exhibiting periodicity or seasonality, such as groceries and beauty products. While Section 4.7 validated the overall generalizability of the model's recommendation performance across domains using public Amazon datasets, this evaluation was limited to aggregate performance and did not examine how the domain-specific periodicity or seasonality of each category interacts with the proposed temporal-context mechanism in detail. A more fine-grained analysis of such domain-specific temporal characteristics is therefore left for future work.

Despite its contributions, this study has several limitations. First, the core experimental dataset (GS SHOP) was confined to a single domain (HFF); while this is supplemented with a generalization check on

public datasets, the primary analysis remains grounded in a single proprietary domain. Second, additional context information, such as user profiles and product metadata, was not fully incorporated beyond temporal features. Third, because the model relies solely on interaction sequences without user or item metadata, it is not directly applicable to cold-start scenarios involving new users or new items with no prior interaction history. Fourth, extending the temporal granularity beyond Month/Day/Day of week (e.g., to hour- or minute-level features) would increase the concatenation dimensionality linearly and may introduce greater data sparsity per categorical bin, potentially requiring continuous encodings (e.g., sinusoidal or Time2Vec) rather than learned categorical embeddings. Future studies should evaluate the model across diverse e-commerce domains and integrate static user and item features with sequential behavior to improve personalization. We expect that employing such contextual information will help facilitate more precise and personalized recommendation experiences in real-world service environments.

**Acknowledgement:** GS Retail has made joint efforts to develop the broadcasting and communication industry through the linkage development of industry-academic cooperation and the development of new technologies such as AI.

**Funding Statement:** This research was funded by GS Retail Co., Ltd.

**Author Contributions:** Conceptualization: Duseo Pak, Jiho Kim and Hongchul Lee; methodology: Duseo Pak; software: Duseo Pak and Jiho Kim; validation: Duseo Pak, Jiho Kim and Hongchul Lee; formal analysis: Duseo Pak; investigation: Duseo Pak and Jiho Kim; resources: Hongchul Lee; data curation: Duseo Pak and Hongchul Lee; writing—original draft preparation: Duseo Pak; writing—review and editing: Duseo Pak, Jiho Kim, and Hongchul Lee; visualization: Duseo Pak, Jiho Kim and Hongchul Lee; supervision: Jiho Kim and Hongchul Lee; project administration: Jiho Kim and Hongchul Lee. All authors reviewed and approved the final version of the manuscript.

**Availability of Data and Materials:** Data is not available due to legal and commercial restrictions. The data underlying this study are not publicly available as they contain personal click information of users of a commercial platform and are subject to privacy and confidentiality restrictions.

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Ko H, Lee S, Park Y, Choi A. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*. 2022;11(1):141. doi:10.3390/electronics11010141.
2. Schafer JB, Konstan JA, Riedl J. E-commerce recommendation applications. *Data Min Knowl Discov*. 2001;5(1):115–53. doi:10.1023/A:1009804230409.
3. Sun F, Liu J, Wu J, Pei C, Lin X, Ou W, et al. BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*; 2019 Nov 3–7; Beijing, China. New York, NY, USA: ACM; 2019. p. 1441–50. doi:10.1145/3357384.3357895.
4. Kang WC, McAuley J. Self-attentive sequential recommendation. In: *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*; 2018 Nov 17–20; Singapore. New York, NY, USA: IEEE; 2018. p. 197–206. doi:10.1109/ICDM.2018.00035.
5. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-based recommendations with recurrent neural networks. In: *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*; 2016 May 2–4; San Juan, PR, USA.
6. Tay Y, Dehghani M, Bahri D, Metzler D. Efficient transformers: a survey. *ACM Comput Surv*. 2022;55(6):1–28. doi:10.1145/3530811.

7. Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans Neural Netw.* 1994;5(2):157–66. doi:10.1109/72.279181.
8. Boka TF, Niu Z, Neupane RB. A survey of sequential recommendation systems: techniques, evaluation, and future directions. *Inf Syst.* 2024;125(5):102427. doi:10.1016/j.is.2024.102427.
9. Liu C, Lin J, Wang J, Liu H, Caverlee J. Mamba4rec: towards efficient sequential recommendation with selective state space models. In: *Proceedings of the 2nd International Workshop on Resource-Efficient Learning for Knowledge Discovery (RelKD 2024) at KDD'24*; 2024 Aug 25; Barcelona, Spain. New York, NY, USA: ACM; 2024.
10. Fang H, Zhang D, Shu Y, Guo G. Deep learning for sequential recommendation: algorithms, influential factors, and evaluations. *ACM Trans Inf Syst.* 2020;39(1):1–42. doi:10.1145/3426723.
11. Kang W-S, Jeong H-J, Lee S, Choi S-M. Independent representation of side information for sequential recommendation. *IEEE Access.* 2024;12:148516–24. doi:10.1109/ACCESS.2024.3476976.
12. Zhang T, Zhao P, Liu Y, Sheng VS, Xu J, Wang D, et al. Feature-level deeper self-attention network for sequential recommendation. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*; 2019 Aug 10–16; Macao, China. Menlo Park, CA, USA: AAAI Press; 2019. p. 4320–6. doi:10.5555/3367471.3367642.
13. Zhou K, Wang H, Zhao WX, Zhu Y, Wang S, Zhang F, et al. S3-rec: self-supervised learning for sequential recommendation with mutual information maximization. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*; 2020 Oct 19–23; Virtual. New York, NY, USA: ACM; 2020. p. 1893–902. doi:10.1145/3340531.3411954.
14. Elahi E, Anwar S, Al-Kfairy M, Rodrigues JJ, Nguetilbaye A, Halim Z, et al. Graph attention-based neural collaborative filtering for item-specific recommendation system using knowledge graph. *Expert Syst Appl.* 2025;266(1):126133. doi:10.1016/j.eswa.2024.126133.
15. Patro SGK, Mishra BK, Panda SK, Kumar R, Long HV, Taniar D. Cold start aware hybrid recommender system approach for E-commerce users. *Soft Comput.* 2023;27(4):2071–91. doi:10.1007/s00500-022-07378-0.
16. Donkers T, Loepp B, Ziegler J. Sequential user-based recurrent neural network recommendations. In: *Proceedings of the 11th ACM Conference on Recommender Systems*; 2017 Aug 27–31; Como, Italy. New York, NY, USA: ACM; 2017. p. 152–60. doi:10.1145/3109859.3109877.
17. Li J, Wang Y, McAuley J. Time interval aware self-attention for sequential recommendation. In: *Proceedings of the 13th International Conference on Web Search and Data Mining*; 2020 Feb 3–7; Houston, TX, USA. New York, NY, USA: ACM; 2020. p. 322–30. doi:10.1145/3336191.3371786.
18. Gu A, Goel K, Ré C. Efficiently modeling long sequences with structured state spaces. In: *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*; 2022 Apr 25–29; Virtual.
19. Gu A, Dao T, Ermon S, Rudra A, Ré C. HiPPO: recurrent memory with optimal polynomial projections. *Adv Neural Inf Process Syst.* 2020;33:1474–87. doi:10.48550/arxiv.2008.07669.
20. Smith JT, Warrington A, Linderman SW. Simplified state space layers for sequence modeling. In: *Proceedings of the 11th International Conference on Learning Representations (ICLR 2023)*; 2023 May 1–5; Kigali, Rwanda.
21. Gu A, Dao T. Mamba: linear-time sequence modeling with selective state spaces. In: *Proceedings of the 1st Conference on Language Modeling*; 2024 Oct 7–9; Philadelphia, PA, USA.
22. Liu Z, Liu Q, Wang Y, Wang W, Jia P, Wang M, et al. SIGMA: selective gated mamba for sequential recommendation. In: *Proceedings of the 39th Annual AAAI Conference on Artificial Intelligence*; 2025 Feb 25–Mar 4; Philadelphia, PA, USA. Menlo Park, CA, USA: AAAI Press; 2025. p. 12264–72. doi:10.1609/aaai.v39i12.33336.
23. Zhang Q, Wen H, Yuan W, Chen C, Yang M, Yiu S-M, et al. Hmamba: hyperbolic mamba for sequential recommendation. *ACM Trans Inf Syst.* 2026;44(5):1–27. doi:10.1145/3811405.
24. GS SHOP [Internet]. 2002 [cited 2026 Aug 26]. Available from: <https://www.gsshop.com/index.gs>.
25. Latifi S, Jannach D, Ferraro A. Sequential recommendation: a study on transformers, nearest neighbors and sampled metrics. *Inf Sci.* 2022;609(2):660–78. doi:10.1016/j.ins.2022.07.079.
26. Zhu X, Li L, Liu W, Luo X. Multi-level sequence denoising with cross-signal contrastive learning for sequential recommendation. *Neural Netw.* 2024;179(6):106480. doi:10.1016/j.neunet.2024.106480.
27. Zhao WX, Mu S, Hou Y, Lin Z, Chen Y, Pan X, et al. RecBole: towards a unified, comprehensive and efficient framework for recommendation algorithms. In: *Proceedings of the 30th ACM International Conference on*

- Information & Knowledge Management; 2021 Nov 1–5; Virtual. New York, NY, USA: ACM; 2021. p. 4653–64. doi:10.1145/3459637.3482016.
28. Deshpande M, Karypis G. Item-based top-N recommendation algorithms. *ACM Trans Inf Syst.* 2004;22(1):143–77. doi:10.1145/963770.963776.
  29. Järvelin K, Kekäläinen J. Cumulated gain-based evaluation of IR techniques. *ACM Trans Inf Syst.* 2002;20(4):422–46. doi:10.1145/582415.582418.