

RWKV4Rec: RWKV-Based Personalized Sequential Recommendation Model

MENGWEI YUAN, LINKAI WAN, ZENGMIN XU, and ZIYUAN XU, School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, China and Center for Applied Mathematics of Guangxi (GUET), Guilin, China

WEIJIAN RUAN, The Smart City Research Institute of China Electronics Technology Group Corporation, Shenzhen, China

Sequential recommender systems, as a core technology for personalized services, have been widely applied across various domains including e-commerce and social networks. Transformer-based sequential recommendation models face challenges such as high resource consumption and inefficient sequential processing, limiting their deployment and real-time recommendation performance. Inspired by the efficiency of the RWKV architecture in handling long sequences in natural language processing, this article proposes RWKV4Rec (RWKV for Sequential Recommendation), the first application of RWKV to sequential recommendation tasks. Our RWKV4Rec model leverages RWKV's efficient long-sequence processing capability and linear computational complexity, reducing resource demands and eliminating additional sequential processing operations. We design an item-RWKV block module and propose a Low-Rank Time Mix based on LoRA technology, which adaptively assigns weights to historical items at each timestep to generate expressive behavior sequence features. Combined with explicit user embeddings and personalized local item matrices, it enhances sequence modeling for efficient and accurate interest prediction. Experiments on four benchmark datasets demonstrate that RWKV4Rec outperforms state-of-the-art methods, achieving 1.80–3.76% improvements in NDCG@10.

CCS Concepts: • **Applied computing** → **Electronic commerce**;

Additional Key Words and Phrases: Sequential recommender system, RWKV4Rec, Linear computational complexity, Low-Rank Time Mix, Partial item Embedding

Associate Editor: Gabriele Tolomei

This research was supported in part by the Guangxi Natural Science Foundation under Grants 2024GXNSFAA010493 and 2025GXNSFAA069661, the National Natural Science Foundation of China under Grants 62402365 and 61862015, and the Science and Technology Project of Guangxi under Grants AD23023002 and AD21220114.

Authors' Contact Information: Mengwei Yuan, School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, China and Center for Applied Mathematics of Guangxi (GUET), Guilin, China; e-mail: ymw@mails.guet.edu.cn; Linkai Wan, School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, China and Center for Applied Mathematics of Guangxi (GUET), Guilin, China; e-mail: wuliike@outlook.com; Zengmin Xu (corresponding author), School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, China and Center for Applied Mathematics of Guangxi (GUET), Guilin, China; e-mail: xzm@guet.edu.cn; Ziyuan Xu, School of Mathematics and Computing Science, Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin, China and Center for Applied Mathematics of Guangxi (GUET), Guilin, China; e-mail: 1600510132@mails.guet.edu.cn; Weijian Ruan, The Smart City Research Institute of China Electronics Technology Group Corporation, Shenzhen, China; e-mail: rweij@whu.edu.cn.

*For correspondence, please contact Zengmin Xu and Weijian Ruan.



This work is licensed under Creative Commons Attribution International 4.0.

© 2026 Copyright held by the owner/author(s).

ACM 1556-472X/2026/6-ART92

<https://doi.org/10.1145/3810245>

ACM Reference format:

Mengwei Yuan, Linkai Wan, Zengmin Xu, Ziyuan Xu, and Weijian Ruan. 2026. RWKV4Rec: RWKV-Based Personalized Sequential Recommendation Model. *ACM Trans. Knowl. Discov. Data.* 20, 6, Article 92 (June 2026), 27 pages.
<https://doi.org/10.1145/3810245>

1 Introduction

Sequential recommender systems [50] predict that content users might be interested in their content on the basis of their past behavior sequences. For instance, e-commerce platforms can recommend products aligned with a user's browsing or purchasing history to increase conversion rates; social media platforms can suggest posts or accounts based on a user's interaction history; music and video streaming services can recommend songs or videos similar to those in a user's playlists; online education platforms can suggest courses or learning resources tailored to a user's learning path and progress; travel booking websites can recommend destinations or accommodations on the basis of a user's search intentions; and gaming platforms can suggest relevant games or quests on the basis of gameplay data and achievements. These systems have become integral to daily life, enhancing the user experience and satisfaction across multiple domains.

The core challenge of sequential recommendation models lies in understanding and predicting how user behavior patterns and preferences evolve. Consequently, capturing these higher-order dynamics succinctly is a primary focus in this field. Traditional methods, such as Markov chain models, can describe a series of cascading effects but are limited in capturing short-term dependencies between user behaviors [36]. In contrast, modern sequential models such as **Recurrent Neural Networks (RNNs)** [27], Long Short-Term Memory Networks [18], **Gated Recurrent Units (GRUs)** [7], and more advanced architectures like the Transformer [48] and its variants [9], demonstrate the ability to handle long sequence data efficiently. The Transformer architecture, with its self-attention mechanism, eliminates traditional recursive paths, enabling the efficient parallel processing of sequence data and significantly enhancing the model's ability to understand long-range dependencies. Sequential modeling ideas have also been explored in other application domains, such as visual tracking tasks, further reflecting the broad applicability of sequence modeling methods [39, 40]. However, sequence recommendation models based on the Transformer architecture face several challenges:

- As time progresses, user behavioral sequences grow longer, causing the model to consider more historical information, which leads to high memory consumption and computational resource requirements due to the quadratic complexity of the Transformer [21, 61].
- Positional information is often critical when processing sequence data [12, 42], especially in recommendation tasks where the order of the user's behavioral sequence reflects temporal relationships. Since the self-attention mechanism does not include recurrent or convolutional modules, positional information is typically added to the input vectors through Positional Encoding.
- According to the working principle of recommender systems, when a user's interest is predicted at time $t + 1$, the model should consider only the pre- t items. However, the output of self-attention contains embeddings of subsequent information, which requires an attention mask [48] to prevent data leakage by modifying the attention calculation to ignore future information.

The resource-intensive nature and the absence of inductive biases present significant limitations, constraining deployment in certain scenarios and necessitating continuous optimization strategies.

In recent years, the widespread adoption of **Large Language Models (LLMs)** such as ChatGPT has renewed interest in the Transformer architecture and spurred ongoing exploration of its core structure. Before the emergence of the Transformer, RNNs dominated the field of language modeling; however, their training processes were challenging to parallelize effectively, severely limiting the scale of the models. In 2023, Bloomberg et al. proposed a novel language model algorithm—RWKV [33], which combines the advantages of both Transformers and RNNs to balance computational efficiency and expressive power in neural networks. RWKV naturally aggregates and controls information across input channels through time-shift operations and employs trainable position-aware weight decay vectors to capture relative positional information between elements in a sequence. This approach reduces the quadratic computational complexity of the self-attention mechanism to a linear level, demonstrating superior performance potential over Transformers in long-sequence reasoning scenarios.

Inspired by RWKV's efficiency in handling long sequences and its capability for processing sequential logic, we adapt it for sequential recommendation tasks. Addressing the challenge of parameter redundancy and the risk of overfitting in sequential recommendation, we introduce a Low-Rank Time Mix mechanism inspired by **Low-Rank Adaptation (LoRA)** [19]. While data sparsity does not mathematically imply a low-rank structure, recent theoretical findings indicate that the intrinsic dimension of user preferences is significantly lower than the high-dimensional observation space [20, 45]. Consequently, full-rank parameter updates often fit high-frequency noise rather than the true underlying manifold. By imposing low-rank constraints on weight updates, our model leverages the strong regularization effect of low-rank factorization [23, 24], which has been proven to tighten generalization bounds and filter out noise. Furthermore, contrary to the traditional view that LoRA is limited to fine-tuning, recent advancements such as ReLoRA [25] and GaLore [65] demonstrate that low-rank architectures are highly effective for training models from scratch, enabling efficient exploration of the parameter space while maintaining representational power. In summary, this article makes the following contributions:

- *First application*: This is the first application of RWKV to sequence recommendation tasks, addressing the inefficiencies of traditional models concerning resource consumption and sequential processing.
- *RWKV4Rec model*: We introduce RWKV4Rec, a sequence recommendation model based on the RWKV architecture. RWKV4Rec capitalizes on the efficient long-sequence processing capabilities and linear complexity of RWKV, thereby reducing computational overhead and eliminating the need for additional sequential handling operations (such as Positional Encoding and Attention Mask). The personalized settings enhance the model's ability to process user behavior sequences, while a negative sampling strategy guides the training process.
- *Item-RWKV block module*: We designed an item-RWKV block module. Leveraging the low intrinsic dimensionality of user preferences, we design an item-RWKV block module and propose a Low-Rank Time Mix based on LoRA technology. This mechanism not only reduces computational overhead but also serves as a structural regularizer to filter out noise from sparse interactions, thereby generating expressive behavior sequence features. The module adaptively assigns weights to historical items at each time step, generating behavior sequence feature representations that are information-rich and expressive.
- *Superior performance and valuable indication*: Comparative experiments were conducted using four benchmark datasets to evaluate RWKV4Rec against state-of-the-art methods, validating its effectiveness as a more efficient solution for sequence recommendation tasks. Through these experiments, we identify the model's application challenges and delineate areas for improvement in future work.

2 Related Work

2.1 Sequential Recommendation Models

Sequential recommender systems center around user behavior sequences, learning the evolution of user preferences through the temporal patterns of interactions with items to achieve precise and personalized dynamic recommendations. Compared with traditional recommendation methods [1] (content-based, collaborative filtering [26, 30, 31, 38, 41, 57, 64], and hybrid approaches [2, 6]) sequential recommendation delves deeper into the order and context of behavior, integrates extensively into social media, e-commerce, and video platforms, and drives significant growth in the internet economy. As an important branch of recommender systems [54], the study of sequential recommendation models has evolved alongside advancements in machine learning and deep learning.

In early explorations of sequential recommendation, researchers attempted to use Markov chain models, such as FPMC [37] and Fossil [15], to capture the temporal dependencies in user behavior. However, Markov chain models typically consider only first-order or finite-order state transitions, which limits their ability to capture complex dependencies in long sequences, thus constraining the performance of Markov chain-based sequential recommendation models when dealing with intricate user behavior sequences. With the rise of deep learning, models based on RNNs and CNNs have begun to be applied to sequential recommendation. These models can handle sequences of arbitrary length and learn long-term dependencies, indicating that a significant leap forward for a sequential recommendation. GRU4Rec [17], introduced in 2015, was the first successful application of the GRU in conversational recommendation, significantly improving recommendation accuracy. Caser [47] used CNNs to capture local patterns in sequences while using RNNs to capture global order information. NextItNet [60] employed CNNs with dilated convolutions to capture sequence dependencies while avoiding the sequential computation bottleneck of RNNs. However, the inherent limitations of neural network architectures—such as the sequential dependency of RNNs, which restricts parallelization, and the fixed kernel size of CNNs, which requires additional handling for sequences of varying lengths—imply that in practical applications, improvements to RNN- or CNN-based sequential recommendation models must be made in conjunction with other techniques.

The emergence of the Transformer model, with its powerful parallel processing capability and effective capture of long-range dependencies, quickly achieved breakthroughs in natural language processing. Subsequently, this model was introduced into the domain of sequential recommendation. Representative works include SASRec [49], which was the first to apply the self-attention mechanism to sequential recommendation. It is capable of capturing global dependencies in user behavior sequences, significantly advancing the development of sequential recommendation models. Bert4Rec [46] drew inspiration from the widely acclaimed BERT model in natural language processing and adopted a bidirectional sequence modeling approach to capture rich contextual semantic information in sequential data. AttRec [63] leveraged the self-attention mechanism to model item-to-item relationships in users' historical interactions, thereby better learning representations of users' short-term interests. It also employed metric learning to model long-term preferences, taking into account both "local" and "global" perspectives. The SSE-PT model [53], proposed in 2020, recognizes the importance of user embeddings in sequence modeling and incorporates a "Stochastic Shared Embeddings" regularization technique, demonstrating its superior performance. In 2021, Liu et al. [28] introduced a non-invasive self-attention mechanism that effectively leverages side information within the BERT framework to generate improved attention distributions. DuoRec [34], proposed in 2022, employs contrastive learning techniques to address representation degeneration in sequential recommendation, thereby enhancing recommendation performance.

and accuracy. In the same year, Xie et al. presented CL4SRec [55] using a contrastive learning framework, which constructs self-supervised signals through three data augmentation methods to more effectively encode user representations. In 2023, Du et al. developed the Frequency-Enhanced Hybrid Attention Network (FEARec) [10], which shifts the perspective from the time domain to the frequency domain to address the limitations of current self-attention mechanisms in capturing high-frequency information. Ye et al. proposed a simple yet effective Graph Masked Autoencoder for sequential recommendation (MAERec) [59], which adaptively and dynamically extracts global item interaction information for self-supervised enhancement. Most recently, in 2024, BSARec [43] was proposed. This work first revealed the low-pass filtering and over-smoothing issues caused by the self-attention mechanism in sequential recommendation. By introducing Fourier transforms to inject perceptual bias, the model successfully addressed the over-smoothing problem through the integration of both low- and high-frequency information. Although Transformer models have achieved remarkable success in natural language processing and other sequential tasks, their high memory consumption, quadratic computational complexity, and large model capacity still impose limitations on the application scenarios of Transformer-based sequential recommendation models.

In recent years, LLMs have demonstrated remarkable semantic understanding and generation capabilities within natural language processing, leading to their widespread adoption across various fields of AI. In **Sequential Recommendation Systems (SRSs)**, LLMs offer innovative solutions to key challenges, including personalized recommendations, explainable recommendations, and diversity recommendations. In 2024, Zhu et al. proposed CLLM4Rec [67], the first generative recommendation system to tightly integrate LLM paradigms with ID (Item-ID) paradigms, addressing the semantic gap between natural language and recommendation tasks. That same year, Wei et al. introduced LLMRec [52], which enhances recommendation system performance through three simple yet effective LLM-based graph augmentation strategies. Zhang et al. developed LC-Rec [66], a model that better integrates linguistic and collaborative semantics to directly generate recommended items from entire item sets without relying on candidate items. In 2025, Chen et al. proposed DLCRec [5], which achieves fine-grained diversity control in LLM-based recommendation systems through task decomposition strategies for sequential reasoning. The SLMRec by Xu et al. [56] explored distilling LLMs into smaller ones, offering a feasible solution for deploying SRSs. Currently, LLM-based recommendation systems have become a key focus for both academia and industry. However, this direction still faces technical challenges regarding model efficiency, deployment costs, and domain adaptation, and its practical application requires further exploration and refinement.

2.2 RWKV

RWKV is an efficient method for sequence modeling that combines the parallel computing advantages of Transformers with the sequence order processing capabilities of RNNs, enhancing the efficiency and effectiveness of models in long-sequence inference scenarios. The design of RWKV draws inspiration from an **Attention Free Transformer (AFT)** [61], a novel neural network architecture introduced by Apple Inc. in 2021. By employing techniques such as residual connections to eliminate the attention mechanism, AFT effectively reduces computational complexity while enhancing model performance, as shown in Figure 1.

$$Attn^+(W, K, V)_t = \frac{\sum_{i=1}^t e^{w_{t,i} + k_i} \cdot v_i}{\sum_{i=1}^t e^{w_{t,i} + k_i}}. \quad (1)$$

In this formulation, t denotes the position currently being computed for output; i serves as the traversal index, iterating over sequence positions from 1 to t ; $k_i \in \mathbb{R}^d$ (where d represents the

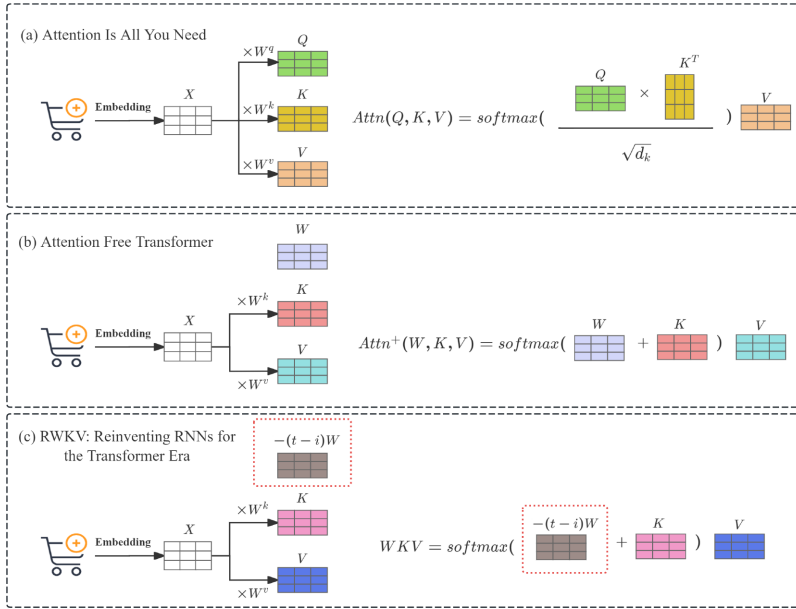


Fig. 1. Comparison of the different attention mechanisms. (a) The traditional attention mechanism calculation depends on the similarity of Query-Key, and the computational complexity increases quadratically with the sequence length. (b) The graph is the computational expression of AFT and reduces the computational complexity to linear complexity by introducing a trainable pair-by-pair position bias matrix W . Figure (c) shows the calculation method of the RWKV model, which uses a special time mixing mechanism by multiplying each element in W with information that decreases in relative position starting from the current time point. This article establishes a sequence recommendation model based on RWKV, which also inherits the advantage of linear complexity. AFT, Attention Free Transformer.

hidden dimension) denotes the Key vector at position i ; $v_i \in \mathbb{R}^d$ represents the Value vector at position i ; $W \in \mathbb{R}^{T \times T}$ (where T is the sequence length) constitutes a learnable pairwise positional bias matrix, representing the most fundamental innovation in AFT. Each element $w_{t,i}$ is a scalar that specifically learns the relationship strength between position t and position i . In practical implementations, low-rank factorization or other parameterization methods are typically applied to W to reduce parameter count. The core conceptual significance of this formulation lies in its decoupling of Query-Key interactions from Value computations.

In RWKV, the author reconstructs each element $w_{t,i}$ of the position bias matrix W as a channel-wise decaying time vector. During computation, these time-decay vectors are multiplied by relative positional information that decays as it looks back from the current time point, forming a unique temporal mixing mechanism:

$$w_{t,i} = -(t - i)w. \quad (2)$$

Here, t represents the index of the current timestep, i represents the index of a previous timestep, and w is a weight coefficient used to adjust the degree of time decay, typically a positive number. This allows the model to capture relative positional information between elements in the sequence, reducing the time complexity of traditional self-attention mechanisms to a linear level, achieving highly parallelized training and efficient inference and demonstrating superior performance potential compared with Transformers in long-sequence inference scenarios.

The introduction of the RWKV provides a new approach to address the issues of resource intensiveness and the lack of an explicit sequential processing mechanism in Transformer-based sequential recommendation models. This article proposes a sequence recommendation model, RWKV4Rec, on the basis of the RWKV architecture, which possesses RWKV's efficient processing capabilities for long sequence data. In sequence recommendation tasks, the model can adaptively assign weights to historical items in the user behavior sequence at each timestep, resulting in rich feature representations that effectively capture user behavior patterns. This leads to accurate predictions of user interests, personalized recommendations, and efficient utilization of computational resources.

2.3 Application of Low-Rank Methods

Datasets in recommender systems are often highly sparse. Taking the User-Item interaction matrix as an example, most users interact with only a small number of items, resulting in the majority of elements in the matrix being zero. This sparsity indicates that the intrinsic dimensionality of the data (i.e., the dimensionality of effective information) is usually much lower than its apparent dimensionality. **Matrix Factorization (MF)** [31, 57] is an important technique in recommender systems that addresses the issue of data sparsity through low-rank approximation. MF is primarily used to handle collaborative filtering problems [26]. Its core idea is to decompose the User-Item interaction matrix into the product of two low-dimensional matrices, thereby uncovering latent user preferences and item features. Specifically, for a User-Item rating matrix $R \in \mathbb{R}^{m \times n}$ (with m is users, n is items, and R is a sparse matrix), it is decomposed into two low-dimensional matrices:

$$R \approx U \cdot V^T, \quad (3)$$

where $U \in \mathbb{R}^{m \times k}$ represents the user latent feature matrix (each row corresponds to a k -dimensional latent vector for a user); and $V \in \mathbb{R}^{n \times k}$ represents the item latent feature matrix (each row corresponds to a k -dimensional latent vector for an item); k is the number of latent factors after dimensionality reduction, and $k \ll m, n$. Commonly used matrix decomposition algorithms include Singular Value Decomposition and Alternating Least Squares. In practical applications, MF methods are widely adopted in various recommendation scenarios due to their simplicity and efficiency.

While MF focuses on data approximation, recent advancements in deep learning address parameter efficiency. LoRA [19] was introduced as a parameter-efficient alternative to full fine-tuning for large-scale models. It is motivated by the hypothesis that task-specific weight updates often lie in a low-dimensional subspace. Concretely, for a weight matrix $W_0 \in \mathbb{R}^{d \times h}$, LoRA parameterizes the update as $\Delta W = BA$, where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times h}$ with $r \ll \min(d, h)$. Training typically freezes W_0 and optimizes only the low-rank factors; the learned update can be merged into the base weights at inference time ($W = W_0 + BA$), avoiding additional latency.

Beyond the conventional pre-training/fine-tuning paradigm, a growing body of work studies low-rank structure as an explicit inductive bias in *training* and *optimization* more broadly. This line is conceptually distinct from classical MF methods mentioned above: whereas MF factorizes the *interaction matrix* to address sparsity, low-rank parameterizations constrain the *model-side update space*, specifically parameter increments, within neural networks. From a theoretical standpoint, recent rank- and Schatten-norm-aware analyses indicate that approximate low-rank structure in neural-network weight matrices can lead to capacity control and generalization bounds that scale with rank-related quantities under suitable assumptions [23]. Earlier work on collaborative prediction with low-rank matrices also provides related background in recommendation-oriented settings [44], although it studies matrix prediction rather than neural-network parameterization. From an optimization perspective, modern training dynamics can also exhibit implicit regularization effects; for instance, sharpness-aware training in scale-invariant settings has been shown to bias solutions

toward more structured regimes with improved generalization behavior [24]. In recommender systems and collaborative filtering, geometric views similarly posit that user–item interactions admit low-dimensional latent structure, motivating methods that exploit geometry or manifold-like assumptions to improve robustness under sparse observations [20, 29]. Finally, recent efforts such as ReLoRA [25] and GaLore [65] demonstrate the practicality of training-time low-rank updates or projections as mechanisms for improving memory and computational efficiency while maintaining competitive performance.

Taken together, these developments indicate that low-rank mechanisms may serve as a useful structural bias beyond parameter-efficient fine-tuning, particularly in settings where model capacity, efficiency, and data sparsity jointly constrain learning, including sequential recommendation.

3 Proposed Approach

In this section, we introduce the proposed sequential recommendation model RWKV4Rec, which is based on the RWKV architecture. We retain the strengths of the original RWKV architecture and make necessary modifications to flexibly apply it to sequential recommendation tasks. Section 3.1 presents the overall framework of the proposed RWKV4Rec model; Section 3.2 details the internal submodules of the item-RWKV block; Section 3.3 describes the optimization strategy for personalized sequential recommendations; and Section 3.4 explains the model training process and the loss function used.

In the sequential recommendation setting, let $U = \{u_1, u_2, \dots, u_{|U|}\}$ (where $|U|$ represents the total number of users) denote a set of users, and $I = \{i_1, i_2, \dots, i_{|I|}\}$ (where $|I|$ represents the total number of items) be the set of items. The action sequence $S_u = (s_1^u, s_2^u, \dots, s_{|S_u|}^u)$ (where $|S_u|$ represents the number of action sequences) represents the chronological sequence of interactions performed by user $u \in U$, where $s_t^u \in I$ denotes the item interacted with by user u at timestep t . During training, given the historical sequence of actions S_u of user u , the model predicts the next item that the user might interact with, denoted as:

$$P(s_{|S_u|+1}^u = i | S_u). \quad (4)$$

3.1 RWKV4Rec Framework

The overall framework of the RWKV4Rec model proposed in this article is shown in Figure 2(a), consisting of an embedding layer, L stacked item-RWKV blocks, an output layer, and a prediction layer.

Consistent with conventional processing in SRSs, the user’s behavioral sequence $(s_1^u, s_2^u, \dots, s_{|S_u|-1}^u)$ is first preprocessed to generate fixed-length sequences $s = (s_1, s_2, \dots, s_n)$, where n represents the maximum sequence length set for the recommendation model. If the length of the user’s behavior sequence exceeds n , only the most recent n behaviors are considered; if the sequence length is less than n , constant “0” padding is added on the left until the sequence reaches length n . Through the model’s embedding layer, the embedded matrix of the user’s behavior sequence is obtained as $E \in \mathbb{R}^{n \times d}$, where d is the latent dimension.

In RWKV4Rec, Time Shift linearly combines the current timestep’s input $E_t \in \mathbb{R}^{n \times d}$ and the previous timestep’s output $E_{t-1} \in \mathbb{R}^{n \times d}$ with weights, generating a new state vector $\mu E_t + (1 - \mu)E_{t-1}$. This implicitly includes positional information in the internal state updates. The decay rate is controlled by a learnable parameter vector, and the older the information is, the smaller its weight.

The item-RWKV block adopts a stack of sequence encoders similar to the Transformer design, with each block comprising a Low-Rank Time Mix and a Channel Mix module, as shown in Figure 3. Low-Rank Time Mix serves as an alternative to traditional self-attention mechanisms,

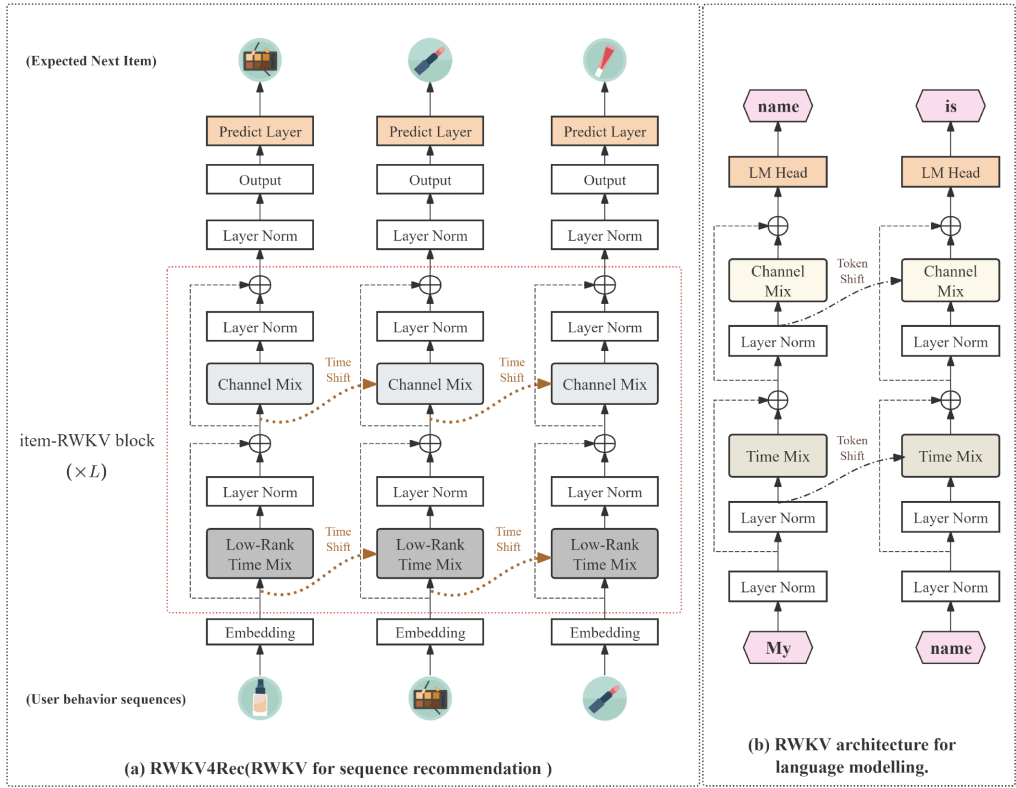


Fig. 2. Comparison of the overall architecture of RWKV4Rec and the RWKV architecture for LLMs. (a) The RWKV4Rec model consists of an embedding layer, multiple item-RWKV blocks, an output layer, and a prediction layer. Each item-RWKV block generates feature representations of the user's behavioral sequence. Inside each item-RWKV block, there is a Low-Rank Time Mix and a Channel Mix module, which use residual connections and time shifts to naturally aggregate and control information in the input channels, aiding the model in adapting to sequence data. (b) The RWKV architecture for LLMs is also composed of multiple RWKV blocks, followed by a language modeling head that outputs the probability distribution of the next token.

performing global attention computations with linear complexity. Channel Mix operates similarly to a feedforward network, taking the output from Low-Rank Time Mix layer as input to process and mix information from different features, passing the mixed features to the next layer or Low-Rank Time Mix layer. Through L stacked item-RWKV blocks, adaptive weighting is assigned to each historical item in the user's behavior sequence $E_t^{(L)} \in \mathbb{R}^{n \times d}$, producing a feature representation for each sequence position. To alleviate overfitting and improve model stability, **Layer Normalization (LayerNorm)** [4] is applied to both Low-Rank Time Mix and Channel Mix modules:

$$\text{LayerNorm}(x) = \gamma \left(\frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \right) + \beta. \quad (5)$$

Among these, γ is the scale factor, and β is the shift factor, both of which are trainable parameters; μ is the mean of the input vector x , σ is the standard deviation of the input vector x , and ϵ is a small constant used to avoid division by zero. Residual connections [14] are established between the input to each item-RWKV block and each normalization layer, preventing the vanishing gradient

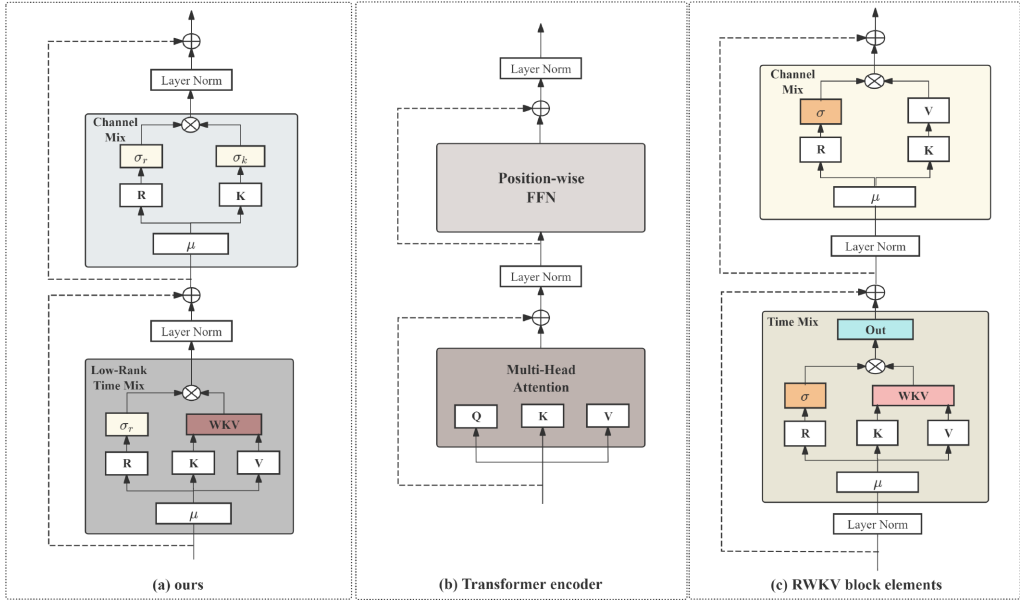


Fig. 3. Comparison of the item-RWKV block used in this model with the Transformer encoder and the original RWKV block. (a) This figure illustrates Low-Rank Time Mix and Channel Mix modules inside the item-RWKV block. Adjustments are made to the position of LayerNorm and certain processing within the submodules, enabling the model to effectively capture user behavior patterns. (b) This figure shows the structure of the Transformer encoder, which includes multi-head attention layers and feedforward network layers. (c) This figure depicts the internal submodule structure of the original RWKV block. The item-RWKV Block used in this article is improved upon this basis.

problem during training and enabling effective low-level features to be more easily captured by the model.

The output layer of the model consists of a linear layer, which receives the output from the last layer of multiple blocks, producing the feature representation of the user's behavior for subsequent loss function calculations and predictions. The prediction layer is part of the entire model architecture responsible for generating the final recommendation results, primarily predicting the items or actions that the user might be interested in next on the basis of their historical behavior sequence.

3.2 Item-RWKV Block

Based on the RWKV framework, this article proposes the item-RWKV block, which is adapted for sequential recommendation tasks and inherits RWKV's efficient long-sequence processing and linear time complexity. By employing Low-Rank Time Mix and Channel Mix modules, the model generates behavior sequence feature representations that are information-rich. Considering the sparsity characteristics of user behavior data, LayerNorm, and residual connections are incorporated to improve model stability, prevent overfitting, and alleviate the problem of gradient vanishing.

3.2.1 Low-Rank Time Mix. The primary objective of the Low-Rank Time Mix module is to capture temporal dependencies within user behavior sequences while maintaining computational efficiency. Traditionally, adapting LLMs involves updating full-rank weight matrices, which is computationally

expensive and prone to overfitting, especially when applied to sequential recommendation tasks characterized by high-dimensional but sparse interactions.

We propose a Low-Rank Time Mix mechanism that imposes a low-rank constraint on the weight update matrices. It is important to clarify the distinction between the rank of the user-item interaction matrix and the rank of the parameter updates. While sparse interaction data does not strictly imply a low-rank structure in the mathematical sense, recent studies on the geometry of recommendation spaces suggest that user preferences are governed by a limited number of latent factors, indicating a low *intrinsic dimensionality* [20, 45]. Therefore, enforcing a low-rank structure on parameter updates (ΔW) serves as a crucial regularization technique. Recent theoretical and optimization studies [23, 24] suggest that low-rank constraints can reduce effective model complexity and act as a form of structural regularization, which may be helpful in sparse and noisy settings. Furthermore, although LoRA [19] was originally designed for fine-tuning, recent work such as ReLoRA [25] and GaLore [65] has demonstrated that low-rank parameterization is also highly effective for training models from scratch, as it encourages the learning of compact and robust representations.

In this section, inspired by the low-rank update mechanism of LoRA, we propose a Low-Rank Time Mix method. This approach significantly reduces the number of trainable parameters during training while preserving the capability of temporal sequence modeling. The working principle of Low-Rank Time Mix module is as follows:

First, the LoRA is defined, where the weight matrix of a linear layer is updated using low-rank matrices A and B :

$$W_r = W_{r_0} + \Delta W_r, \quad (6)$$

$$\Delta W_r = B_r \times A_r, \quad (7)$$

where $W_{r_0} \in \mathbb{R}^{n \times d}$, $B_r \in \mathbb{R}^{n \times r}$, $A_r \in \mathbb{R}^{r \times d}$, and the rank r is much smaller than both n and d .

Recent theory on rank-aware generalization in neural networks provides some support for using low-rank parameterizations as a form of structural regularization. In particular, Ledent et al. [23] show that approximate low-rank structure in neural-network weight matrices can be incorporated into generalization bounds through Schatten- p quasi-norms, yielding a complexity characterization that interpolates between norm-based and parameter-counting regimes. Related earlier work on collaborative prediction with low-rank matrices [44] also reflects the broader connection between low-rank structure and generalization in recommendation-related problems, although it focuses on matrix prediction rather than neural-network parameterization. Motivated by these observations, we treat the low-rank update ΔW in our model as a structural bias that restricts the parameter-update space and may improve robustness under sparse and noisy interactions, rather than as a setting-specific theoretical guarantee.

When the rank r of LoRA is chosen to align with the intrinsic complexity of the recommendation dataset, the low-rank update can retain much of the representational capacity of full-parameter updates. The LoRA layers are injected into the linear weight matrices W_r , W_k , and W_v within the Low-Rank Time Mix module. During output computation, the low-rank matrices are added to the initialized weight matrix parameters to produce the final output.

A trainable weight matrix W_r , passed through the LoRA layer, projects the new state vector produced at a decay rate μ_r into an acceptance matrix $R_t \in \mathbb{R}^{n \times d}$, which measures the degree of acceptance of past information. This function is similar to the forget gate in RNN [27], which filters out unnecessary historical information.

$$R_t = W_r \cdot (\mu_r E_t + (1 - \mu_r) E_{t-1}). \quad (8)$$

The new state vectors produced at different decay rates μ_k are μ_v projected into key vectors $K_t \in \mathbb{R}^{n \times d}$ and value vectors $V_t \in \mathbb{R}^{n \times d}$ through linear transformations. Similarly, W_k and W_v are also obtained through the LoRA layers.

$$K_t = W_k \cdot (\mu_k E_t + (1 - \mu_k) E_{t-1}), \quad (9)$$

$$V_t = W_v \cdot (\mu_v E_t + (1 - \mu_v) E_{t-1}). \quad (10)$$

wkv_t represents the relevance of the user's behavior information at time t relative to the behavior information at previous time steps. This is achieved through a trainable pairwise position bias matrix W , which captures and encodes the relative positional information between the elements in the sequence. To avoid any potential degradation of W , a vector U that focuses solely on the current token is introduced.

$$wkv_t = \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}. \quad (11)$$

The acceptance matrix R_t , after passing through an activation function (e.g., Sigmoid [11], ReLU [13], GELU [16]), is multiplied with wkv_t to obtain the output E'_t at position t . The trainable weight matrix W_o dynamically adjusts the weights at each timestep in the sequence. The computation of E'_t contains historical information, and as t increases, E'_t will depend on increasingly longer historical information.

$$E'_t = W_o \cdot (\sigma(R_t) \cdot wkv_t). \quad (12)$$

When the training data for a recommender system is from a dense dataset, it is recommended to use the original linear layers in the Time Mix module without LoRA-based low-rank adaptation. This is because dense datasets typically have a higher intrinsic dimensionality. If the rank r used in LoRA is much smaller than this intrinsic dimension, critical information in the user behavior sequences may be lost during the low-rank update process, thereby weakening the model's ability to represent and process sequential features. The ablation study results in Section 4.7 also demonstrate that when the recommendation model is trained on dense datasets using Low-Rank Time Mix, the model performance drops significantly.

3.2.2 Channel Mix. The Channel Mix module is another core component that mixes the information obtained by the model across different feature dimensions at each timestep to enhance the model's expressiveness for input data features.

Receiving the output of Time Mix, Channel Mix also requires a time shift to compute new reception vectors $R'_t \in \mathbb{R}^{n \times d}$ and key vectors $K'_t \in \mathbb{R}^{n \times d}$ and applies activation functions to introduce non-linearity. Through a combination of linear transformations W'_v and nonlinear transformations $\sigma(K'_t)$, features are extracted from the input signals, and the intensities of these features are further modulated by the R'_t gating mechanism, ultimately generating the output $E_t^{(l)}$:

$$R'_t = W'_r \cdot (\mu'_r E'_t + (1 - \mu'_r) E'_{t-1}), \quad (13)$$

$$K'_t = W'_k \cdot (\mu'_k E'_t + (1 - \mu'_k) E'_{t-1}), \quad (14)$$

$$E_t^{(l)} = \sigma(R'_t) \cdot (W'_v \cdot \sigma(K'_t)). \quad (15)$$

This design helps the model flexibly capture complex dependencies in the sequence data during processing.

3.2.3 Adjustment of LayerNorm. As a model architecture in the LLM domain, the original RWKV structure performs LayerNorm operations after the sequence data passes through the embedding layer, followed by Time Mix and Channel Mix modules, as shown in Figure 3(c). This design is related to the characteristics of the corpora received by LLMs, such as text sentence length and density, requiring normalization to reduce differences in feature scales of the training data. In recommendation tasks, the training data typically consists of structured data specific to a particular domain with less variation. Based on this characteristic, this article refers to the design of the original Transformer model [48] to adjust the position of LayerNorm. Specifically, the input sequence first passes through Low-Rank Time Mix and Channel Mix modules before normalization, accelerating convergence. Additionally, since Low-Rank Time Mix and Channel Mix modules include nonlinear processing, this arrangement helps maintain consistency in the input distribution, thereby improving the efficiency and stability of model training.

3.3 Optimization for Personalized Sequence Recommendations

3.3.1 User Embedding. To enhance the model's personalization capabilities, we consider using User Embedding to learn users' specific preferences [15, 17, 37, 47]. According to all user behaviors, an explicit user embedding is inserted into the last layer of the item-RWKV block:

$$\hat{E}_t^{(L)} = E_t^{(L)} + U_u, \quad (16)$$

where $U_u \in \mathbb{R}^d$ is the user embedding matrix. However, the comparative experimental results in Section 4.6 indicate that adding an explicit User Embedding does not improve performance.

3.3.2 Partial Item Embedding. The prediction layer computes the dot product between the user behavior feature representation $E_t^{(L)}$ output by the item-RWKV block and the total item embedding $E_I \in \mathbb{R}^{|I| \times d}$ for all items in the set $I = \{i_1, i_2, \dots, i_{|I|}\}$, resulting in predicted scores (logits) for each item. Higher logits indicate a higher match probability, which can be used for recommendation after sorting.

$$\text{logits}_t = E_t^{(L)} \cdot E_I. \quad (17)$$

To reduce the model size and avoid massive computations for all user-item pairs and leverage the sparsity characteristics of user behavior data, this article adopts a strategy from the official code of [49] to establish a lightweight partial item matrix (Partial item Embedding). For each user u , a list of p ($p < |I|$) unrated items is randomly generated, which represents the item embedding matrix $E_p \in \mathbb{R}^{p \times d}$, which includes the next item that truly interacted with in the user's behavior sequence. The prediction score (logit) for the next item by user u is expressed as:

$$\tilde{\text{logits}}_t = E_t^{(L)} \cdot E_p. \quad (18)$$

3.4 Model Training

This article follows the same preprocessing procedures as those in [49]. For all datasets, the interacting items (such as ratings, reviews, purchases, or views) by users are considered positive samples and are sorted chronologically. For each user u , given the sequence of user actions $S_u = (s_1^u, s_2^u, \dots, s_{|S_u|}^u)$, the last action $s_{|S_u|}^u$ is used as the test data, and the second-to-last action $s_{|S_u|-1}^u$ is used as the validation set, and the remaining actions are used for training. During training, for the interaction history of user u , at timestep t , the model predicts the next item $s_{|S_u|+1}^u, s_{|S_u|+2}^u \in I$ on the basis of the previous t items.

The model employs a negative sampling strategy [47, 49, 60] for contrastive learning. For negative sample sequences, i.e., for each item in the user's behavior sequence, the model randomly generates

Table 1. Dataset Information

Datasets	Users	Items	Avg. Length	Actions	Density
Beauty	52,024	57,289	5.63	0.4 M	0.01%
Video	31,013	23,715	7.26	0.3 M	0.03%
Steam	334,730	13,047	9.02	3.7 M	0.07%
ml-1m	6,040	3,416	163.5	1.0 M	4.79%

m items that the user has not interacted with at time t from the item set $I = \{i_1, i_2, \dots, i_{|I|}\}$. The positive and negative sample sequences are transformed into their corresponding feature representations $E_{pos} \in \mathbb{R}^{n \times d}$ and $E_{neg_j} \in \mathbb{R}^{n \times d}$, $j \in (1, 2, \dots, m)$ through the item embedding layer. These representations are then dot-produced with the user behavior features to compute the corresponding logit scores, which are used in the loss function calculation:

$$logits_{pos_t} = E_t^{(L)} \cdot E_{pos}, \quad (19)$$

$$logits_{neg_{j,t}} = E_t^{(L)} \cdot E_{neg_j}. \quad (20)$$

A binary cross-entropy loss function [47, 49, 60] is used to compare the model's predicted scores with the actual items clicked by users, guiding the training process and enabling the model to better predict subsequent user interests:

$$Loss = - \sum_{S^u \in S} \sum_{t \in [1, 2, \dots, n]} \left[\log(\sigma(logits_{pos_t})) + \sum_{j \notin S^u} \log(1 - \sigma(logits_{neg_{j,t}})) \right], \quad (21)$$

where S is the set of all user behavior sequences, σ is the sigmoid function, j represents a negative sample, and padding items in the sequence data need to be ignored during actual computation.

4 Experiments

4.1 Datasets

Experiments were conducted on four publicly available benchmark datasets, which differ significantly in terms of domain, platform, and sparsity. Table 1 summarizes the main statistics of these datasets.

Amazon Datasets [32]: These datasets record user reviews of items on Amazon.com, compiled by multiple research institutions and universities. The latest version, maintained by the University of California, San Diego, is Amazon Review Data (2018), which includes 233 million reviews between May 1996 and October 2018. These datasets cover various categories, ranging from books and electronics to clothing and home goods, making them important resources for research in recommender systems, natural language processing, sentiment analysis, and other fields. In this article, we selected the “Beauty” and “Video games” categories. The “Beauty” category contains ratings from 52,024 users for 57,289 beauty products, whereas the “Video Games” category includes ratings from 31,013 users for 23,715 video games. This dataset is characterized by its large scale, rich diversity, and long temporal span, with high sparsity and variability.

Steam Dataset: This is a widely used collection of game-related data, primarily composed of Steam platform user behavior and game metadata. The dataset was collected and organized by researchers and data scientists from the public Steam API and user profiles. It typically includes user profiles, game information, purchase records, playtime, and other behaviors, aiming to facilitate research and development in areas such as game recommender systems, user behavior analysis,

Table 2. Comparison of the Computational Complexity of the Models

Model	Computational Complexity
GRU4Rec (ICLR2016) [17]	$O(\mathfrak{L} \cdot T \cdot d^2 + k \cdot d)$
Caser (WSDM2018) [47]	$O(n \cdot T \cdot d \cdot h + m \cdot T \cdot d + k \cdot d)^a$
SASRec (ICDM2018) [49]	$O(\mathfrak{L} \cdot T^2 \cdot d + \mathfrak{L} \cdot T \cdot d^2 + k \cdot d)$
BERT4Rec (CIKM2019) [46]	$O(\mathfrak{L} \cdot T^2 \cdot d + \mathfrak{L} \cdot T \cdot d^2 + k \cdot d)$
Attrec (arXiv2018) [63]	$O(T^2 \cdot d + T \cdot d^2 + k \cdot d)$
DuoRec (WSDM2022) [34]	$O(\mathfrak{L} \cdot T^2 \cdot d + \mathfrak{L} \cdot T \cdot d^2 + k \cdot d)$
CL4SRec (ICDE2022) [55]	$O(\mathfrak{L} \cdot T^2 \cdot d + \mathfrak{L} \cdot T \cdot d^2 + k \cdot d)$
FEARec (SIGIR2023) [10]	$O(\mathfrak{L} \cdot T \cdot \log T \cdot d + \mathfrak{L} \cdot T \cdot d^2 + k \cdot d)$
MAERec (SIGIR2023) [59]	$O(\mathfrak{L} \cdot T^2 \cdot d + \mathfrak{L} \cdot T \cdot d^2 + k \cdot d)$
BSARec (AAAI2024) [43]	$O(\mathfrak{L} \cdot T^2 \cdot d + \mathfrak{L} \cdot T \cdot d^2 + k \cdot d)$
RWKV4Rec (Ours)	$O(\mathfrak{L} \cdot T \cdot d + T \cdot d)$

Here, $\mathfrak{L}$ is the number of model encoders or layers, T is the length of the sequence received by the model, d is the dimension of the hidden layer, and k is the number of items to be recommended.

^aFor the Caser model, n is the number of horizontal filters, h is the height of the horizontal filters, and m is the number of vertical filters.

and market trend prediction. The data used in this article include 334,730 users and 13,047 games, spanning from October 2010 to January 2018.

MovieLens Dataset: The MovieLens dataset is a well-known collection of movie ratings gathered and organized by GroupLens Research. It contains user ratings and viewing records for movies, making it valuable for researching and developing recommender systems. By analyzing user ratings and viewing habits, personalized recommendation algorithms can be created. This article utilizes the ml-1m benchmark dataset released in February 2003, which consists of 1 million ratings from 6,040 users for 3,416 movies.

The two Amazon datasets have the shortest average length of user behavior sequences among the datasets, the Steam dataset has the largest number of behavior sequences, and the ml-1m dataset is the densest.

4.2 Computational Complexity Comparison

Table 2 compares the computational complexity of the proposed method with that of neural network-based and self-attention-based sequential recommendation models. The self-attention mechanism in traditional Transformers can capture dependencies between any two positions in a sequence, resulting in a time complexity of $O(T^2 \cdot d)$ for computing the attention matrix $QK^\top$, where L is the sequence length. During training, the self-attention mechanism can fully leverage the parallel computing capabilities of GPUs. Therefore, self-attention-based sequential recommendation models such as SASRec [49] and AttRec [63] are typically very efficient in short-sequence recommendation tasks. However, in long-sequence scenarios, their computational cost increases significantly due to the quadratic complexity. In contrast to models like AttRec and SASRec, RWKV4Rec captures temporal dependencies through a recursive mechanism based on RWKV's Time Shift approach, reducing the quadratic computational complexity of self-attention to linear with respect to the input sequence length $O(T \cdot d)$. Combined with LoRA technology, RWKV4Rec significantly reduces the demand for computational resources in long-sequence recommendation scenarios, demonstrating the effectiveness of the proposed method in addressing resource-intensive problems.

During training, RWKV4Rec processes the entire sequence by constructing two sequences using the shift-by-one method from paper [26]. These two sequences differ by one position and can be mixed in parallel. The core computation in Low-Rank Time Mix module is wkv_t , which involves a formula that is essentially similar to Softmax, except that an additional term is added to both the numerator and the denominator. Both are still weighted sums. Therefore, RWKV4Rec supports parallel computation just like the Transformer, enabling efficient training.

4.3 Evaluation Metrics

To evaluate the recommendation effectiveness of all the models, this article uses the **Hit Rate (HR)** [3], **Normalized Discounted Cumulative Gain (NDCG)** [51], and **Mean Reciprocal Rank (MRR)** [8] as evaluation metrics.

HR@k. It focuses on whether the recommendation list includes items of interest to the user, measuring whether the user's next interaction item appears in a recommendation list of length k . This metric emphasizes the "accuracy" of the prediction. The calculation formula is as follows:

$$HR@k = \frac{1}{k} \sum_{i=1}^k hits(i), \quad (22)$$

where k is the number of recommended items, $hits(i)$ indicates whether the value visited by the user's i th interacted target item is included in the model's recommendation list: if it is, the value is 1; otherwise, it is 0. HR directly reflects the recommender system's ability to identify items that users are truly interested in within the recommendation list. However, relying solely on the HR has certain limitations. Specifically, HR only focuses on whether the recommendation list contains items of interest to users, without considering their specific ranking positions. This means that even if a user's truly interested item appears at the end of the recommendation list, as long as that item exists in the list, the HR will still be considered a successful hit. To more comprehensively evaluate the effectiveness of recommendations, it is usually necessary to combine other evaluation metrics that take ranking factors into consideration.

NDCG@k. In recommender systems, NDCG not only considers the relevance of recommended items to user interests, but also emphasizes their ranking positions in the recommendation list, highlighting the "ranking importance." Specifically, $NDCG@k$ is used to evaluate the ranking performance of correctly predicted user interaction items within a recommendation list of length k . The calculation formula is as follows:

$$DCG@k = \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2(i + 1)}, \quad (23)$$

$$IDCG@k = \sum_{i=1}^k \frac{2^{rel(ideal)_i} - 1}{\log_2(i + 1)}, \quad (24)$$

$$NDCG@k = \frac{DCG@k}{IDCG@k}. \quad (25)$$

The core concept of $DCG@k$ (Discounted Cumulative Gain) is that users derive significantly greater satisfaction when preferred items are ranked higher in the recommendation list, compared to lower positions. If relevant results appear in lower ranks, the list score should be penalized, with the magnitude of the penalty being inversely related to the rank of relevant results. Here, $rel_i \in [0, 1]$ represents the relevance score. $IDCG@k$ (Ideal DCG) denotes the maximum achievable DCG value under ideal conditions, where $@(ideal)$ indicates that results are sorted in descending order of relevance, forming a set comprising the top- k results. In other words, it represents the

optimal ranking arrangement. NDCG is obtained by dividing the original DCG by IDCG, yielding a normalized value bounded between 0 and 1.

MRR@k. In recommendation systems, MRR emphasizes the rank position of the user's desired item within the model's recommendation list, where higher rankings (closer to the top) correspond to better performance. The calculation formula is as follows:

$$DCG@k = \frac{1}{U} \sum_{i=1}^{|U|} \frac{1}{rank_i}, \quad (26)$$

where $|U|$ denotes the number of users, $rank_i$ represents the position of the true is the visited value of the i th user in the recommendation list; if the value does not exist in the recommendation list, then $rank_i \rightarrow \infty$.

By comprehensively utilizing these three metrics, we can thoroughly evaluate the model's performance in recommendation tasks. Higher values of both *HR@k*, *NDCG@k* and *MRR@k* indicate that the model possesses superior recommendation capabilities.

4.4 Implementation Details

The hardware platform used for the experiments in this study consists of 512 GB of memory and one NVIDIA 3060 GPU. The software platform consists of Python 3.9 and PyTorch 1.12.1 framework. The training parameters were configured consistently with [49], using the Adam optimizer [22] with a learning rate set to 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. The model was trained for a total of 200 epochs with a batch size of 128. For dense data, the maximum sequence length n was set to 200, using the original linear Time Mix layer without LoRA low-rank approximation. For sparse data, the maximum sequence length n was configured as 50, employing Low-Rank Time Mix with the LoRA layer's rank $r = 4$. The model parameters were taken from [26], with the number of blocks L set to 12, the number of heads h set to 8, and the dimension of each head set to 8. All other parameters were trainable. The performance of the variants and different hyperparameters are examined in the following sections.

4.5 Comparison Experiments

4.5.1 Model Performance. To verify the effectiveness of the proposed method, this article compares it with the following representative baselines: non-sequential recommendation with BPR [35], which does not consider sequences; sequence recommendation methods based on neural networks, including GRU4Rec [17] and Caser [47]; and sequence recommendation methods based on self-attention mechanisms, such as Attrec [63], SASRec [49], BERT4Rec [46], DuoRec [34], CL4SRec [55], FEARec [10], MAERec [59], and BSARec [43].

LLM-based sequential recommendation models typically have large parameter sizes (e.g., CLLM4Rec with 7 billion parameters [67]), resulting in significantly higher training and deployment costs compared to conventional methods. Moreover, the generative nature of LLMs introduces inference latency, which must be considered for real-time recommendation requirements. Therefore, this article does not include LLM-based sequential recommendation models in the baseline performance comparison.

To ensure the fairness and reproducibility of the comparative experiments, all baseline models were implemented based on the official code released by the original authors, with uniform adaptations applied to guarantee that all comparative models and our proposed model are evaluated under identical experimental environments, data preprocessing procedures, and evaluation protocols, obtaining the results shown in Table 3.

Table 3. Comparison of Recommendation Model Performance

Models	Beauty			Video			Steam			ml-1m		
	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR	HR	NDCG	MRR
BPR (UAI2009) [35]	0.3731	0.2484	0.2119	0.4610	0.2816	0.2387	0.7729	0.4965	0.4263	0.5692	0.3034	0.2350
GRU4Rec (ICLR2016) [17]	0.4135	0.2183	0.1800	0.5869	0.2953	0.2395	0.8153	0.5559	0.4774	0.7430	0.5158	0.4379
Caser (WSDM2018) [47]	0.4574	0.2831	0.2468	0.5985	0.3606	0.3010	0.8235	0.5836	0.5049	0.7873	0.5295	0.4423
Attrec (arXiv2018) [63]	0.4292	0.2724	0.2349	0.5731	0.3328	0.2781	0.8318	0.5764	0.4986	0.7837	0.5362	0.4521
SASRec (ICDM2018) [49]	<u>0.4716</u>	0.3055	<u>0.2595</u>	0.6243	0.4124	0.3414	<u>0.8635</u>	0.6056	0.5482	<u>0.8198</u>	0.5870	0.5094
BERT4Rec (CIKM2019) [46]	0.4275	0.2367	0.1880	0.5910	0.3776	0.3139	0.8392	0.5780	0.5036	0.8045	0.5778	0.5103
DuoRec (WSDM2022) [34]	0.4626	0.2876	0.2414	0.6366	0.4210	0.3492	0.8579	0.5942	0.5209	0.8204	<u>0.5887</u>	0.5182
CL4SRec (ICDE2022) [55]	0.4567	0.2952	0.2470	0.6453	0.4237	0.3561	0.8139	0.5554	0.4834	0.7072	0.4179	0.3287
FEARec (SIGIR2023) [10]	0.4628	0.2884	0.2373	0.6120	0.4187	0.3489	0.8175	0.5802	0.5103	0.7116	0.4333	0.3418
MAERec (SIGIR2023) [59]	0.4726	0.3062	0.2578	0.6406	0.4329	<u>0.3765</u>	0.8585	<u>0.6073</u>	<u>0.5382</u>	0.8218	0.5880	0.5189
BSARec (AAAI2024) [43]	0.4379	0.2756	0.2303	<u>0.6496</u>	<u>0.4339</u>	0.3696	0.8292	0.5704	0.4951	0.8107	0.5715	0.5036
RWKV4Rec (Ours)	0.4668	0.3129	0.2717	0.6578	0.4454	0.3785	0.8702	0.6197	0.5511	0.8105	0.5948	0.5254
Improvement	-1.24%	+2.19%	+4.70%	+1.29%	+2.65%	+0.53%	0.76%	+2.04%	+2.40%	-1.39%	+1.04%	+1.25%

The recommendation list length k is set to 10. In each column, the bolded numbers indicate the best result, and the underlined numbers indicate the second-best result for each metric.

The experimental results demonstrate that our proposed RWKV4Rec model achieves superior overall performance compared to all baseline models on three sparse datasets: Beauty, Video, and Steam. The evaluation metrics NDCG@10 reach 0.3129, 0.4454, and 0.6197, respectively, representing approximately 2% improvements over the second-best results. We hypothesize three potential reasons for this performance advantage:

- RWKV4Rec’s recursive mechanism: RWKV4Rec naturally handles the sequential logic in the data through a recursive mechanism similar to RNNs. There is no need to add extra positional encoding to impart positional information to the input data, nor is there a need for attention mask operations to prevent leakage of future information, leading to a more stable training process.
- Short-term dependency patterns: Leveraging its RNN-like recurrent characteristics, RWKV4Rec focuses on capturing short-term dependency patterns within sequences. This inductive bias enables the model to more accurately identify users’ immediate preferences when their interests evolve gradually, thereby ranking the true next interaction item in higher positions. Owing to the logarithmic discounting property of the NDCG metric, it exhibits high sensitivity to ranking improvements. Consequently, NDCG consistently captures the enhancement in ranking precision across all four datasets.
- Enhanced representational power: RWKV4Rec has sufficient network depth and nonlinear processing, which enhances the model’s ability to represent input sequences. The item-RWKV block can capture dependencies between items in the sequence, obtain rich behavioral sequence feature representations, and thereby improve the accuracy of recommendations.

On the dense dataset ml-1m, RWKV4Rec demonstrates slightly inferior overall performance compared to SASRec. Specifically, its HR@10 score of 0.8056 represents a 1.39% decrease relative to the top-performing SASRec model. This performance gap can be attributed to RWKV’s inherent architectural characteristics. Similar to RNNs, RWKV relies on hidden states (denoted as wkv_t in its structure) to memorize historical information, where memory retention exhibits progressive decay over time. Consequently, its capability to capture long-range historical dependencies is less effective than that of Transformer-based models. The observed decline in HR@10 on the sparsest dataset, Beauty, can be attributed to this phenomenon. For binary metrics like HR that solely concern whether items appear within the top- k recommendations, the accuracy diminishes when recommendation sequences are less influenced by long-range historical information.

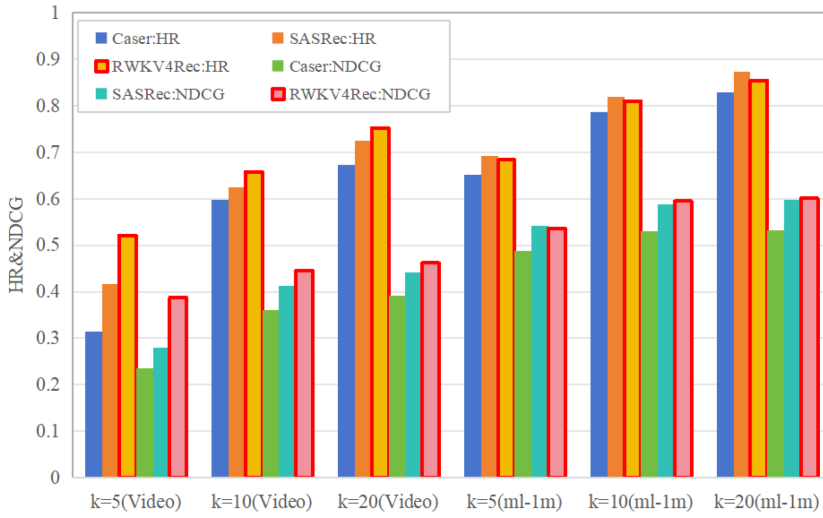


Fig. 4. Impact of the number of recommendations k on model performance across different datasets. For the HR and NDCG metrics, the performance of RWKV4Rec is highlighted with red outlines.

4.5.2 Impact of the Number of Recommendations k . As shown in Figure 4, the number of top recommendations (TOP- K) in a sequential recommender system can influence the evaluation metrics. We chose the sparse dataset Video and the dense dataset ml-1m to study the impact of $k = 5, 10, 20$ on the HR and NDCG metrics for Caser, SASRec, and RWKV4Rec. For all the metrics, higher values indicate better performance.

The experimental results demonstrate that as the recommendation list length k increases, both $HR@k$ and $NDCG@k$ show varying degrees of improvement across the two test datasets (the sparse dataset Video and dense dataset ml-1m). Specifically, on the Video dataset, RWKV4Rec exhibits significant advantages over traditional sequential recommendation models Caser and SASRec across all metrics, with consistently higher $HR@k$ and $NDCG@k$ values. However, on the ml-1m dataset, while RWKV4Rec outperforms Caser overall, it still shows some performance gaps compared to SASRec. Notably, in terms of model efficiency, RWKV4Rec consumes significantly fewer computational resources than the baseline models, allowing it to maintain good recommendation quality while offering higher practical value and deployment feasibility. This balance between performance and efficiency gives RWKV4Rec strong competitiveness in real-world applications, particularly in scenarios with computational constraints, as evidenced by the consistent resource efficiency advantage across different dataset types and the persistent performance differentials between various metrics.

As k increases, all the metrics on both datasets show varying degrees of improvement. On the Video dataset, RWKV4Rec achieves higher HR and NDCG scores than the Caser and SASRec methods do. On the ml-1m dataset, RWKV4Rec outperforms Caser but is inferior to SASRec. However, RWKV4Rec still demonstrates competitive performance, achieving the highest NDCG score among the three models when $k = 10$.

4.5.3 Training Efficiency Comparison. Table 4 compares parameter counts and computational time between our proposed method and self-attention-based sequential recommendation models. On the sparse Video dataset, the RWKV4Rec model based on the RWKV architecture introduces only a marginal increase in parameters while achieving enhanced recommendation performance,

Table 4. Comparative Analysis of Model Parameters and Computational Time

Models	Video			ml-1m		
	NDCG	Params (KB)	Time (s/epoch)	NDCG	Params (KB)	Time (s/epoch)
SASRec (ICDM2018) [49]	0.4124	3165.50	5.54	0.5870	646.88	3.23
BERT4Rec (CIKM2019) [46]	0.3776	3358.25	5.57	0.5778	839.75	3.62
DuoRec (WSDM2022) [34]	0.4210	3358.25	5.62	<u>0.5887</u>	839.75	3.56
CL4SRec (ICDE2022) [55]	0.4237	3358.25	5.94	0.3931	839.75	3.45
FEARec (SIGIR2023) [10]	0.4187	3358.25	15.13	0.4333	839.75	10.09
MAERec (SIGIR2023) [59]	0.4329	6290.50	67.16	0.5880	1234.50	35.13
BSARec (AAAI2024) [43]	<u>0.4339</u>	3166.75	7.78	0.5715	840.38	4.12
RWKV4Rec (Ours)	0.4454	9286.30	14.28	0.5948	3550.57	8.69

The recommendation list length $|k|$ is set to 10. In the NDCG column, the bolded numbers indicate the best result for this metric, and the underlined numbers indicate the second-best result.

demonstrating the effectiveness of our approach. For the dense ml-1m dataset, although our model achieves the highest NDCG score, this comes with a notable increase in parameter count, which is directly related to the number of Item-RWKV blocks L . To capture long-range dependencies in sequences, increasing L expands network depth and enhances semantic representation capacity, albeit at the cost of higher parameterization, indicating certain applicability constraints of our model in recommendation systems. Compared to self-attention-based sequential recommendation models, RWKV4Retains Transformer-like parallel computation capabilities, though its recurrent architecture impacts runtime efficiency. Comprehensive evaluations on both datasets demonstrate that RWKV4Rec delivers stable performance and efficiency, confirming its viability as an efficient model for sequential recommendation.

This computational efficiency analysis further clarifies our model's advantages and applicable scenarios, confirming its potential and feasibility for recommendation tasks. In future work, we will prioritize the development of lightweight attention modules tailored to our architectural framework. The integration of lightweight attention mechanisms with the RWKV architecture can preserve the core advantages and computational speed of traditional self-attention while reducing computational and storage requirements through optimization techniques.

4.5.4 Comparison of Normalization Operations. Figure 5 shows the impact of different normalization operations on model recommendation performance. Normalization is a commonly used preprocessing and regularization technique with several main roles, including reducing the scale difference between training data features, accelerating convergence, enhancing model stability, mitigating the gradient vanishing/explosion problem, and providing regularization effects.

In addition to LayerNorm, this study investigates the impact of RMSNorm [62] and FixedNorm [58] on model performance.

RMSNorm is similar to LayerNorm, but it uses the **Root Mean Square (RMS)** of the input samples to calculate and apply normalization, typically providing better performance and stability in certain cases (such as in Transformer models). FixedNorm is a variant of L2Norm that performs a normalization operation with a fixed scale. It does not have any learnable parameters and normalizes simply by using the L2 norm of the input vector and a fixed scaling factor. This operation is commonly used in deep learning models to help stabilize the training process and accelerate convergence.

Although the calculations for RMSNorm and FixedNorm are relatively simple and can be more stable than those for LayerNorm in certain situations, especially when handling long sequence data

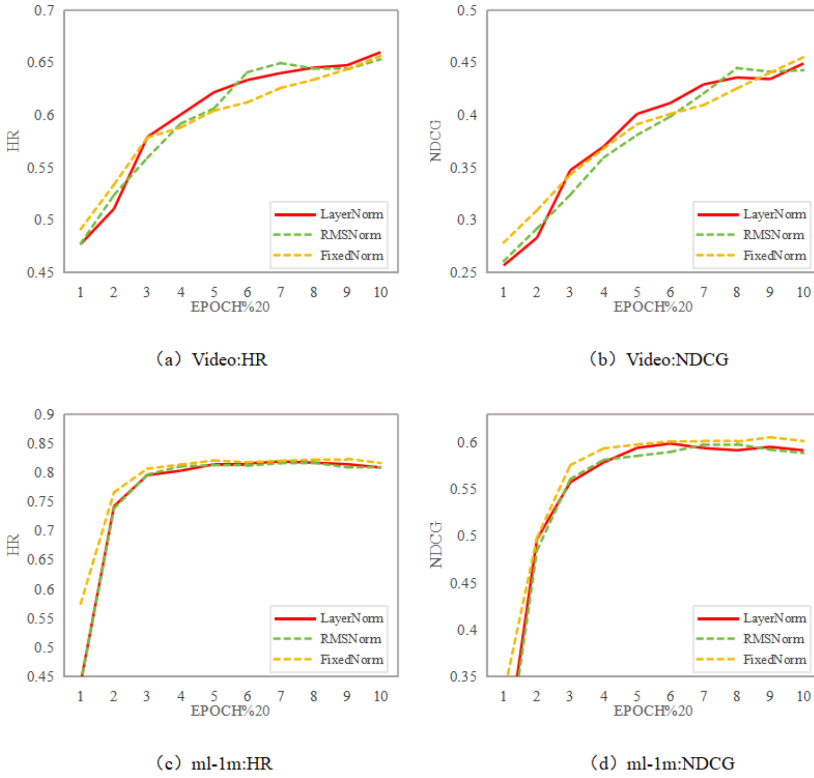


Fig. 5. Impact of different normalization operations on model recommendation performance. Figures (a) and (b) show that LayerNorm achieves the highest performance on sparse datasets compared with RMSNorm and FixedNorm. Figures (b) and (d) show that FixedNorm is more stable than LayerNorm on dense datasets and can serve as an efficient alternative, improving training speed and model performance for longer sequences and larger model sizes.

because they are not affected by the statistical characteristics of specific positions in the sequence, LayerNorm is preferred for its flexibility and broad applicability in the model architecture presented in this article. In specific scenarios, RMSNorm and FixedNorm can serve as efficient alternatives, such as when dealing with long sequences and larger model sizes, to increase training speed and model performance.

4.6 Hyperparameter Experiment

The item-RWKV block module in RWKV4Rec is a crucial part of the model. It generates the feature representation of each user's behavior sequence, which forms the basis for personalized recommendations.

1. Experimental results demonstrate a significant correlation between the number of item-RWKV blocks L and model performance, where L determines the model's representation capacity for input sequences. Specifically, increasing L leads to deeper network layers and richer semantic expressive power, consequently improving the effectiveness of the sequential recommendation model. In hyperparameter experiments with the item-RWKV block number L set to 6, the evaluation metric NDCG@10 achieved 0.3805 on the sparse dataset Video and 0.5691 on the dense dataset ml-1m. Compared to the model's default configuration $L = 12$, reducing L degraded model performance,

Table 5. Hyperparameter Experiment for the Model Architecture (NDCG@10)

Datasets	Video	ml-1m
Default	0.4454	0.5948
$L=6$	0.3805	0.5691
$r=2$	0.3967↓	0.3120↓
$r=8$	0.4429	0.4863↓

A dash “↓” indicates a significant performance drop (greater than 10%). The model’s default setting for the number of item-RWKV blocks L is 12, and the rank of the Low-Rank Time Mix module r is 4.

with particularly noticeable deterioration on sparse datasets. These results empirically demonstrate that increasing network depth remains an effective approach for enhancing recommendation performance under reasonable conditions, especially when handling complex user behavior patterns.

2. The rank hyperparameter r in the Low-Rank Time Mix module acts as a critical control for the tradeoff between *model capacity* and *regularization*. As shown in Table 5, setting an appropriate rank is essential for optimal performance. On the dense ml-1m dataset, reducing r to 2 causes a drastic performance drop in NDCG@10 (from 0.5948 to 0.3120). This indicates a capacity bottleneck, where the subspace is too constrained to capture the complex, high-fidelity interaction patterns inherent in dense data, leading to underfitting. Conversely, increasing r to 8 also degrades performance (0.4863) compared to the optimal setting ($r = 4$). This suggests that simply increasing capacity does not guarantee better results; instead, a specific low-rank constraint ($r = 4$) serves as a necessary structural regularizer, forcing the model to learn more compact and robust representations. Similarly, on the sparse Video dataset, deviation from the optimal rank (yielding 0.3967 at $r = 2$ and 0.4429 at $r = 8$) leads to suboptimal results compared to the default (0.4454), confirming that the Low-Rank Time Mix must be tuned to match the intrinsic complexity of the recommendation task.

4.7 Ablation Study

Given the numerous components in our proposed model architecture, we conduct ablation studies to analyze their contributions. Table 6 presents the performance comparison of different model variants on both the sparse dataset Video and dense dataset ml-1m.

Low-Rank Time Mix vs. Time Mix with the Original Linear Layer. The results demonstrate that on the sparse dataset Video, the LoRA-based Low-Rank Time Mix achieves an NDCG@10 score of 0.4490. This approach not only reduces the number of trainable parameters but also enhances temporal sequence modeling capabilities, confirming the effectiveness of low-rank parameter updates. Moreover, incorporating the LoRA layer is computationally efficient, requiring minimal additional memory overhead without introducing any training latency. However, on the dense dataset ml-1m, Low-Rank Time Mix leads to a performance dip compared to the full-rank baseline. Consistent with our analysis in Section 4.6, this can be attributed to the capacity-regularization tradeoff. The dense interactions in ml-1m require a higher model capacity to be fully captured. When the rank is constrained, it creates a capacity bottleneck that may result in slight underfitting on such information-rich datasets, whereas this same constraint acts as a beneficial regularizer on sparse datasets like Video.

Table 6. Ablation Study of the Model Architecture (NDCG@10)

Architecture	Video	ml-1m
Low-Rank Time Mix	0.4490	0.4019↓
Time Mix with the Original Linear Layer	0.4401	0.5985
Standalone Execution of Low-Rank Time Mix	0.2943↓	0.3097↓
Standalone Execution of Time Mix	0.2791↓	0.4120↓
Add User Embedding	0.4176	0.5874
Not Adjust the Position of LayerNorm	0.4053	0.5740
Remove Residual Connections	0.2866↓	0.3206↓
Total Item Embedding	0.4280	0.5891

A dash “↓” indicates a significant performance drop (greater than 10%). Both Low-Rank Time Mix and Time Mix with the original linear layer implementations default to the following architectural configurations: exclusion of User Embedding, adjust the position of LayerNorm, inclusion of residual connections, and adoption of Partial Item Embedding.

Standalone Execution of Low-Rank Time Mix and Time Mix. Experimental results demonstrate that when operating independently, both the Low-Rank Time Mix and Time Mix modules exhibit significant performance degradation across evaluation metrics, observed consistently in both the sparse Video dataset and the dense ml-1m dataset. Time Mix and Channel Mix serve as two core submodules in RWKV, responsible for processing temporal and channel (feature) dimensions in sequential data, respectively. When constructing the main architecture of the RWKV4Rec recommendation model using only $L=12$ stacked Low-Rank Time Mix or Time Mix modules, the absence of complex nonlinear transformations in the feature dimension prevents the model from adequately learning intricate inter-feature relationships. This limitation constrains its expressive capacity and falls substantially short of the performance achieved through collaborative operation of both submodules.

Add User Embedding. The model incorporates user demographic information by using User Embedding for modeling. However, the final experimental results show that it does not improve the model’s performance. The item-RWKV block’s multi-layer temporal modeling architecture comprehensively captures users’ dynamic preferences. The generated sequence representations implicitly encode personalized user information. As a result, the static user features and the behavior sequence features exhibit information overlap, and adding new features leads to redundancy rather than additional value.

Not Adjust the Position of LayerNorm. The experimental results reveal that preserving the original RWKV architecture with normalization preceding Time Mix and Channel Mix modules degrades model performance. The recommender system’s behavioral data exhibits distinct domain-specific structural patterns, where the inherent scale variations themselves carry critical semantic information (e.g., frequently interacted items typically show larger embedding magnitudes). These natural scale differences fundamentally reflect intrinsic behavioral patterns, making explicit LayerNorm operations unnecessary for feature scale unification. Our tests demonstrate that prepending LayerNorm erases distinctive patterns in raw embeddings and reduces the model’s sensitivity to critical behavioral signals.

Remove Residual Connections. The model exhibits significant performance degradation when residual connections are removed, with both key metrics decreasing by over 10%. This deterioration occurs because, in deep network architectures, critical low-level features (e.g., outputs from the

embedding layer and initial residual blocks) fail to propagate effectively to final layers, leading to feature degradation. These foundational features are particularly valuable for recommendation generation. Furthermore, residual connections contribute to model stability—their removal destabilizes the training process and degrades convergence properties.

Total Item Embedding. In the prediction layer, using the Total item Embedding $E_I \in \mathbb{R}^{|I| \times d}$ (where $|I|$ is the total number of items) is used to obtain the prediction score for each item, and items are recommended for sorting. Experimental results show that expanding elements in item embeddings does not enhance recommendation performance, but instead degrades model effectiveness. The embeddings of irrelevant items introduce noise, diluting the discriminative power of key items. In the prediction phase, the time and space complexity of matrix multiplication increases from $O(p \times d)$ to $O(I \times d)$, leading to higher computational costs.

5 Conclusions

This article proposes an RWKV-based sequential recommendation model called RWKV4Rec, aimed at optimizing traditional sequential recommendation models, particularly those based on the Transformer architecture, in terms of resource consumption and sequential processing. RWKV4Rec benefits from strong long-sequence modeling capabilities and natural handling of sequential logic, along with the advantage of linear time complexity. An item-RWKV block module is designed, and by incorporating LoRA technology, we introduce a Low-Rank Time Mix method that reduces computational resource requirements while enhancing the model's ability to understand sequential data. As an efficient solution for sequential recommendation tasks, RWKV4Rec accurately predicts users' subsequent interests by obtaining rich user behavior sequence feature representations. Empirical studies demonstrate that RWKV4Rec performs significantly well on four benchmark datasets, achieving better results than existing recommendation technologies do. However, RWKV4Rec has limitations in remembering long-distance historical information, and future work will focus on expanding the model's ability to handle long-range historical information and further studying the enhancement of personalization in sequential recommender systems as well as addressing the sparsity of user behavior data. Future research can be pursued in the following two directions: (1) Continued optimization of the Item-RWKV Block module: Designing hybrid architectures to explore more effective dynamic weight allocation mechanisms; Incorporating external knowledge graphs (e.g., item attributes, user interest tags) to enhance the model's comprehension of users' long-term interests. (2) Development of lightweight attention modules adapted to the model architecture: Integrating lightweight attention mechanisms with the RWKV framework can preserve the core advantages of traditional self-attention while reducing computational and memory requirements.

Acknowledgements

The authors would like to thank the reviewers and editor for their valuable comments and suggestions.

References

- [1] G. Adomavicius and A. Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734–749.
- [2] D. Agarwal and B. C. Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 19–28.
- [3] A. Alsini, D. Q. Huynh, and A. Datta. 2020. Hit ratio: An evaluation metric for hashtag recommendation. arXiv:2010.01258. Retrieved from <https://arxiv.org/abs/2010.01258>
- [4] J. L. Ba, J. R. Kiros, and G. E. Hinton. 2016. Layer normalization. arXiv:1607.06450. Retrieved from <https://arxiv.org/abs/1607.06450>

- [5] J. J. Chen, C. M. Gao, S. Yuan, S. C. Liu, Q. P. Cai, and P. Jiang. 2025. DLCRec: A novel approach for managing diversity in LLM-based recommender systems. In *Proceedings of the 18th ACM International Conference on Web Search and Data Mining*. ACM, 857–865.
- [6] T. G. Chen, W. N. Zhang, Q. X. Lu, K. Chen, Z. Zheng, and Y. Yu. 2012. SVDFeature: A toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research* 13, 1 (2012), 3619–3622.
- [7] J. Y. Chung, C. Gulcehre, K. Cho, and Y. Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555. Retrieved from <https://arxiv.org/abs/1412.3555>
- [8] N. Craswell. 2009. Mean reciprocal rank. In *Encyclopedia of Database Systems*. Springer, 1703.
- [9] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1, 4171–4186.
- [10] X. Y. Du, H. H. Yuan, P. P. Zhao, J. F. Qu, F. Z. Zhuang, G. F. Liu, and V. S. Sheng. 2023. Frequency enhanced hybrid attention network for sequential recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 78–88.
- [11] S. Elfving, E. Uchibe, and K. Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks* 107 (2018), 3–11.
- [12] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*. MIT Press, 1243–1252.
- [13] X. Glorot, A. Bordes, and Y. Bengio. 2011. Deep sparse rectifier neural networks. *Journal of Machine Learning Research* 15 (2011), 315–323.
- [14] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 770–778.
- [15] R. N. He and J. McAuley. 2016. Fusing similarity models with Markov chains for sparse sequential recommendation. In *Proceedings of the 16th International Conference on Data Mining*. IEEE Computer Society, 191–200.
- [16] D. Hendrycks and K. Gimpel. 2016. Gaussian error linear units (GELUs). arXiv:1606.08415. Retrieved from <https://arxiv.org/abs/1606.08415>
- [17] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*.
- [18] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
- [19] E. J. Hu, Y. L. Shen, P. Wallis, Z. Y. Allen-Zhu, Y. Z. Li, S. Wang, L. Wang, and W. Z. Chen. 2022. LoRA: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations*.
- [20] Hisham Husain and Julien Monteil. 2025. Geometric collaborative filtering with convergence. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*. PMLR, 2728–2736.
- [21] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*. MIT Press, Cambridge, 5156–5165.
- [22] D. P. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.
- [23] Antoine Ledent, Rodrigo Alves, and Yunwen Lei. 2025. Generalization bounds for rank-sparse neural networks. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*. NeurIPS.
- [24] Bingcong Li, Liang Zhang, and Niao He. 2024. Implicit regularization of sharpness-aware minimization for scale-invariant problems. In *Proceedings of the Advances in Neural Information Processing Systems*. Curran Associates, Inc., 44444–44478.
- [25] Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2024. ReLoRA: High-rank training through low-rank updates. In *Proceedings of the 12th International Conference on Learning Representations*. ICLR.
- [26] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.
- [27] Z. C. Lipton, J. Berkowitz, and C. Elkan. 2015. A critical review of recurrent neural networks for sequence learning. arXiv:1506.00019. Retrieved from <https://arxiv.org/abs/1506.00019>
- [28] C. Liu, X. G. Li, G. H. Cai, Z. H. Dong, H. Zhu, and L. F. Shang. 2021. Non-invasive self-attention for side information fusion in sequential recommendation. In *Proceedings of the 15th AAAI Conference on Artificial Intelligence*. AAAI Press, 4249–4256.
- [29] Donald Loveland, Xinyi Wu, Tong Zhao, Danai Koutra, Neil Shah, and Mingxuan Ju. 2025. Understanding and scaling collaborative filtering optimization from the perspective of matrix rank. In *Proceedings of the ACM on Web Conference 2025*. ACM, 436–449.
- [30] Q. Lu, R. L. Wang, J. Zhang, and Y. Ding. 2013. Recommendation algorithm integrating neighborhood model and latent semantic model. *Computer Engineering and Applications* 49, 19 (2013), 100–103 + 134.

- [31] A. Mnih and R. Salakhutdinov. 2007. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*. Curran Associates, Inc., 1257–1264.
- [32] J. Ni, J. Li, and J. McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 188–197.
- [33] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, M. Grella, et al. 2023. RWKV: Reinventing RNNs for the transformer era. arXiv:2305.13048. Retrieved from <https://arxiv.org/abs/2305.13048>
- [34] R. H. Qiu, Z. Huang, H. Z. Yin, and Z. J. Wang. 2022. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. ACM, 813–823.
- [35] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.
- [36] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 811–820.
- [37] S. Rendle, C. Freudenthaler, and L. Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International World Wide Web Conference*. ACM, 811–820.
- [38] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. ACM, 175–186.
- [39] Weijian Ruan, Wu Liu, Qian Bao, Jun Chen, Yuhao Cheng, and Tao Mei. 2019. POINet: Pose-guided ovonic insight network for multi-person pose tracking. In *Proceedings of the 27th ACM International Conference on Multimedia*, 284–292.
- [40] Weijian Ruan, Mang Ye, Yi Wu, Wu Liu, Jun Chen, Chao Liang, Ge Li, and Chia-Wen Lin. 2021. TICNet: A target-insight correlation network for object tracking. *IEEE Transactions on Cybernetics* 52, 11 (2021), 12150–12162.
- [41] B. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, 285–295.
- [42] P. Shaw, J. Uszkoreit, and A. Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 2, 464–468.
- [43] Y. Shin, J. Choi, H. Wi, and N. Park. 2024. An attentive inductive bias for sequential recommendation beyond the self-attention. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI Press, 8984–8992.
- [44] Nathan Srebro, Noga Alon, and Tommi Jaakkola. 2004. Generalization error bounds for collaborative prediction with low-rank matrices. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 17. MIT Press.
- [45] Yixin Su, Wei Jiang, Fangquan Lin, Cheng Yang, Sarah M. Erfani, Junhao Gan, Yunxiang Zhao, Ruixuan Li, and Rui Zhang. 2025. Intrinsic and extrinsic factor disentanglement for recommendation in various context scenarios. *ACM Transactions on Information Systems* 43 (2025), 1–32.
- [46] F. Sun, J. Liu, J. Wu, C. H. Pei, X. Lin, W. W. Ou, and P. Jiang. 2019. ERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 1441–1450.
- [47] J. X. Tang and K. Wang. 2018. Personalized top-N sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. ACM, 565–573.
- [48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates, Inc., 6000–6010.
- [49] C. K. Wang and J. McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining*. IEEE Computer Society, 197–206.
- [50] S. J. Wang, L. Hu, Y. Wang, L. B. Cao, Q. Z. Sheng, and M. Orgun. 2019. Sequential recommender systems: Challenges, progress and prospects. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 6332–6338.
- [51] Y. Wang, L. W. Wang, Y. Z. Li, D. He, T. Y. Liu, and W. Chen. 2013. A theoretical analysis of NDCG type ranking measures. In *Proceedings of the 26th Annual Conference on Learning Theory*, 25–54.
- [52] W. Wei, X. B. Ren, J. B. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang. 2024. LLMRec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. ACM, 806–815.
- [53] L. W. Wu, S. G. Li, J. Cho, and J. Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the 114th ACM Conference on Recommender Systems*. ACM, 328–337.

- [54] L. Xiang. 2012. *Practical Recommendation Systems*. Posts and Telecom Press, Beijing.
- [55] X. Xie, F. Sun, Z. Y. Liu, S. W. Wu, J. Y. Gao, B. L. Ding, and B. Cui. 2022. Contrastive learning for sequential recommendation. In *Proceedings of the 2022 IEEE International Conference on Data Engineering*. IEEE Computer Society, 1259–1273.
- [56] W. J. Xu, Q. T. Wu, Z. J. Liang, J. J. Han, X. Y. Ning, Y. X. Shi, W. F. Lin, and Y. F. Zhang. 2025. SLMRec: Distilling large language models into small for sequential recommendation. In *Proceedings of the 13th International Conference on Learning Representations*.
- [57] Y. Yang, Y. Xiang, and L. Xiong. 2012. Collaborative filtering recommendation algorithm based on matrix factorization and user nearest neighbor model. *Journal of Computer Applications* 32, 2 (2012), 395–398.
- [58] Z. Y. Yang, Y. F. Hu, N. Y. Liang, and J. Lv. 2019. Nonnegative matrix factorization with fixed L2-norm constraint. *Circuits, Systems, and Signal Processing* 38, 7 (2019), 3211–3226.
- [59] Y. W. Ye, L. H. Xia, and C. Huang. 2023. Graph masked autoencoder for sequential recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 321–330.
- [60] F. J. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. ACM, 582–590.
- [61] S. F. Zhai, W. Talbott, N. Srivastava, C. Huang, H. L. Goh, R. X. Zhang, and J. Susskind. 2021. An attention free transformer. In *Proceedings of the 9th International Conference on Learning Representations*.
- [62] B. Zhang and R. Sennrich. 2019. Root mean square layer normalization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates, Inc., 12360–12371.
- [63] S. Zhang, Y. Tay, L. Yao, and A. X. Sun. 2018. Next item recommendation with self-attentive metric learning. arXiv:1808.06414. Retrieved from <https://arxiv.org/abs/1808.06414>
- [64] S. Zhang, W. H. Wang, J. Ford, and F. Makedon. 2006. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, Bethesda MD, 549–553.
- [65] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR, 61121–61143.
- [66] B. W. Zheng, Y. P. Hou, H. Y. Lu, Y. Chen, W. X. Zhao, M. Chen, and J.-R. Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *Proceedings of the 2024 IEEE International Conference on Data Engineering*. IEEE Computer Society, 1435–1448.
- [67] Y. C. Zhu, L. Wu, Q. Guo, L. Hong, and J. Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM Web Conference 2024*. ACM, 3162–3172.

Received 14 July 2025; revised 20 January 2026; accepted 5 April 2026