

Capturing User Interests from Data Streams for Continual Sequential Recommendation

Gyuseok Lee
University of Illinois
Urbana-Champaign
Champaign, IL, USA
gyuseok2@illinois.edu

Hyunsik Yoo
University of Illinois
Urbana-Champaign
Champaign, IL, USA
hy40@illinois.edu

Junyoung Hwang
42dot
Seongnam, Republic of Korea
junyoung.hwang@42dot.ai

SeongKu Kang*
Korea University
Seoul, Republic of Korea
seongkukang@korea.ac.kr

Hwanjo Yu
Pohang University of
Science and Technology
Pohang, Republic of Korea
hwanjoyu@postech.ac.kr

Abstract

Transformer-based sequential recommendation (SR) models excel at modeling long-range dependencies, but suffer from high computational costs and catastrophic forgetting during continuous updates. Although continual learning has been applied to recommendation, existing methods gradually forget long-term user preferences and remain underexplored in SR. In this paper, we introduce **Continual Sequential Transformer for Recommendation (CSTRec)**, which effectively adapt to current interests by leveraging preserved historical knowledge. Its core is **Continual Sequential Attention (CSA)**, a linear attention tailored for continual SR, which partially retain historical knowledge without direct access to prior data. CSA features: (1) *Cauchy-Schwarz Normalization* to stabilize learning over time under uneven user interaction frequencies, and (2) *Collaborative Interest Enrichment* via shared, learnable interest pools to mitigate forgetting. We also introduce a new technique for new user adaptation by transferring historical knowledge from existing users with similar interests. Extensive experiments show CSTRec’s superior performance in both knowledge retention and acquisition.¹

CCS Concepts

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

Keywords

Continual Learning, Sequential Recommendation, Linear Attention

ACM Reference Format:

Gyuseok Lee, Hyunsik Yoo, Junyoung Hwang, SeongKu Kang, and Hwanjo Yu. 2026. Capturing User Interests from Data Streams for Continual Sequential Recommendation. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining (WSDM '26)*, February 22–26, 2026, Boise, ID, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3773966.3777977>

*Corresponding author.

¹Our code is available at https://github.com/Gyu-Seok0/CSTRec_WSDM26.



This work is licensed under a Creative Commons Attribution 4.0 International License. *WSDM '26, Boise, ID, USA*

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2292-9/2026/02
<https://doi.org/10.1145/3773966.3777977>

1 Introduction

Sequential recommendation (SR) has gained prominence in both academia and practical applications by capturing sequential patterns in user behavior to enhance next item prediction [11, 17, 34, 56]. Earlier studies use Markov chains [3, 32] and Recurrent Neural Networks (RNNs) [13, 46] to model temporal dependencies in user behavior sequences. However, both approaches struggle to capture long-range dependencies as sequence length increases [17]. To address this, Transformer-based SR models [10, 17, 35, 47, 58] employ self-attention to focus on the most relevant behaviors regardless of position [38]. With its remarkable performance, Transformer has become the dominant approach in SR.

Nevertheless, applying Transformer-based SR models to real-world applications presents non-trivial challenges. As user behavior sequences are continuously arriving, models should adapt to new information for timely recommendations [4, 7, 24]. A simple approach is to retrain the model using the entire user behavior sequences accumulated along data streams. However, training on the whole sequences is extremely time-consuming [27, 43], due to the quadratic complexity of self-attention with respect to input length [5, 18, 40]. Moreover, this approach is impractical in scenarios where not all historical interactions are accessible (e.g., privacy issues) or where memory constraints prevent storing complete sequences during both training and inference [43, 57]. A more cost-effective alternative is to fine-tune the model using only new sequences. However, such overreliance on transient interests risks forgetting historical user interests that may reemerge later, thereby significantly limiting recommendation accuracy [42, 43, 50].

Continual learning (CL) [20, 21, 25], a well-established approach to updating a model with non-stationary data streams, has been actively studied for recommendation [27, 41, 42, 50, 59]. With the goal of adapting to new data while preserving previously learned knowledge, there are two dominant CL approaches: (1) Regularization-based methods [41, 42, 50] impose constraints on the parameter space to prevent significant changes from previously trained parameters. (2) Replay-based methods [2, 27, 59] store small portions of historical data and reuse them in subsequent training, with external memory being updated over time.

Although existing CL methods retain interests from the *relatively recent* past, they are insufficient to preserve those from the *distant* past, gradually forgetting long-term preferences. Specifically, regularization-based methods rely on the most recently learned parameters to transfer acquired knowledge into the subsequent training process. As this greedy preservation continues, new information gradually dilutes historical knowledge, leading to unavoidable forgetting. Similarly, replay-based methods are constrained by limited memory capacity, which necessitates continually updating stored data with newer interactions. This reduces access to older data, further hindering the retention of historical user interests. Furthermore, most CL methods have focused on non-sequential models (e.g., matrix factorization [14, 55], graph neural networks [41, 42]), leaving their application in SR relatively underexplored.

We propose **Continual Sequential Transformer for Recommendation (CSTRec)**, which continuously updates a transformer-based SR model with non-stationary data streams. CSTRec aims to preserve historical interests and leverage them to adapt to current ones, ultimately capturing the trajectory of user interests over time. To facilitate the preservation of historical knowledge, we borrow the idea of linear attention [5, 18, 40]. Linear attention approximates self-attention by performing linear computations at each position and sequentially accumulating hidden states over time, similar to the hidden state propagation in RNNs. This allows the model to partially retain historical knowledge through parametric memories without direct access to all historical sequences, while emulating the expressive power of self-attention [22, 28].

However, naively applying linear attention to continual SR yields suboptimal results due to two challenges: (1) unstable training, caused by an imbalance in the number of interactions per user along data streams; hidden states accumulate user behaviors over time, resulting in disproportionately large values for active users and small values for less active users. This imbalance leads to uneven magnitudes and updates across users, making optimization highly unstable. (2) inevitable forgetting, caused by the continual update of hidden states with new sequences over time. This causes historical knowledge to be continuously overwritten and gradually forgotten.

As a solution, we introduce **Continual Sequential Attention (CSA)**, a specialized linear attention for continual SR, featuring two novel components: (1) *Cauchy-Schwarz Normalization* to resolve unstable learning. Leveraging the Cauchy-Schwarz inequality, we dynamically adjust the magnitudes of hidden states to address the imbalance caused by disparities in the number of interactions. (2) *Collaborative Interest Enrichment* to alleviate inevitable forgetting. We utilize learnable interest pools that store historical interests. For each user context, we retrieve the most relevant interests from the pools and use them to compensate for forgotten user interests. With CSA, CSTRec not only inherits the strengths of linear attention but also effectively addresses its limitations for continual SR. Furthermore, to facilitate the accommodation of newly joined users, we introduce a new technique called *Pseudo-Historical Knowledge Assignment*. By leveraging the historical knowledge of existing users with similar behavioral patterns, it allows new users to be effectively integrated into CSA computation.

Our contributions are summarized as follows:

- We highlight the challenges of employing transformer-based SR models in non-stationary data streams, which have not been studied well in the previous literature. To the best of our knowledge, we are the first to address these challenges.
- We propose CSTRec equipped with CSA—a linear attention mechanism tailored for continual SR—to effectively retain historical knowledge and acquire current one, thereby capturing the trajectory of user interests over time.
- We validate the effectiveness of CSTRec through comprehensive experiments on real-world datasets and provide in-depth analyses to validate each proposed component.

2 Related Work

Sequential Recommendation (SR). Earlier SR studies [3, 32, 33] employ Markov chains (MC). FPMC [32] bridges matrix factorization and MC for next-basket recommendation, while LME [3] applies metric learning to learn Markov embeddings. With the advent of deep learning, RNNs have been applied in SR [13, 23, 36, 46]. GRU4Rec [13] pioneers Gated Recurrent Units (GRUs), and [36] further improves RNN-based models by augmenting data and addressing data distribution shifts. NARM [23] employs two different GRUs to encode both the sequential pattern and the main interest within a given session. Recently, transformer-based SR models [10, 17, 35, 47, 58] have shown exceptional performance by effectively capturing long-range dependencies via self-attention, thereby becoming dominant in SR. SASRec [17] pioneers the use of self-attention [38], and BERT4Rec [35] applies bidirectional self-attention [8]. SSE-PT [47] addresses the lack of personalization in the Transformer by employing stochastic shared embeddings [48].

However, updating Transformer-based SR models along data streams poses two non-trivial challenges: high training costs and catastrophic forgetting, which remain underexplored. Further study is needed to extend the applicability of SR models to real-world applications with continuously arriving user behavior sequences.

Continual Learning (CL). CL is a well-established research area to update a model along data streams [21, 52, 53]. The goal of CL is to effectively balance knowledge acquisition and retention over time [16, 19, 54]. Recently, CL has been actively studied for recommendation [1, 2, 12, 27, 41, 42, 50, 51, 59] to rapidly adapt to new data while leveraging well-preserved historical knowledge. Two key CL approaches are (1) regularization [20, 25, 41, 42] and (2) experience replay [27, 29, 30, 59].

Regularization-based methods penalize rapid changes to previously trained parameters. For example, LWC-KD [41] introduces contrastive knowledge distillation between the parameters of the previously and currently trained graph neural networks. SAIL-PIW [42] proposes personalized imitation weights to adjust knowledge retention based on user preferences being static or dynamic. On the other hand, replay-based methods store and retrieve historical data from external memory, which is continuously updated. ADER [27] assigns memory slots to each item based on its frequency in data streams and retains (session, target item) pairs whose session features are closest to the average feature vector. ReLoop2 [59] introduces a self-correcting loop that stores mispredicted samples in a non-parametric memory to improve future learning.

However, existing CL methods fall short of retaining long-term user interests and thus suffer gradual forgetting. This is primarily

due to the sequential integration of previously learned knowledge into subsequent training processes. As models continually adapt to non-stationary data streams, historical knowledge becomes diluted by new information and is gradually forgotten, making it difficult to capture long-term user preferences. Furthermore, most prior studies have focused solely on non-sequential models, leaving their application in SR relatively underexplored. Therefore, it is necessary to develop a specialized approach tailored to continual SR.

3 PRELIMINARIES

3.1 Problem Formulation

We view the entire data stream D as consecutive data blocks $[D_1, D_2, \dots, D_t, \dots]$, where D_t contains interaction sequences observed during the time period t (e.g., weekly or monthly). Let \mathcal{U}^t and \mathcal{I}^t be the set of users and items within D_t , respectively. Let $S_u^t = [i_1^t, \dots, i_k^t, \dots, i_{|S_u^t|}^t]$ be the sequence of items with which user u interacted in D_t , where i_k^t denotes the k -th item in S_u^t . Traditional SR aims to predict the next item $i_{|S_u^t|+1}^t$ given S_u^t . Building upon this, our task (i.e., continual SR) is to predict the next item for each incoming interaction sequence (i.e., $S_u^1, S_u^2, \dots, S_u^t, \dots$). Note that at each time period t , we update the model parameters solely on the newly arrived block D_t , without accessing any previous data $D_{<t}$.

3.2 Background

3.2.1 Transformer-based SR model. The architecture of the Transformer-based SR models [17] consists of multi-head attention (MH), a position-wise feed-forward network (FFN), layer normalization, and dropout. Given an input sequence $S_u^t = [i_1^t, \dots, i_N^t]$ of N items, we embed each item into a d -dimensional vector and stack them to form $E_{S_u^t} \in \mathbb{R}^{N \times d}$. $E_{S_u^t}$ serves as the initial hidden states \mathbf{H}^0 , on which the self-attention operation is performed as follows:

$$\text{head}_i = \text{softmax}\left(\frac{(\mathbf{H}^{l-1} \mathbf{W}_Q^{(i)})(\mathbf{H}^{l-1} \mathbf{W}_K^{(i)})^\top}{\sqrt{d}}\right)(\mathbf{H}^{l-1} \mathbf{W}_V^{(i)}),$$

$$\text{MH}(\mathbf{H}^{l-1}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \mathbf{W}_O, \quad (1)$$

$$\mathbf{G}^{l-1} = \text{LayerNorm}(\mathbf{H}^{l-1} + \text{Dropout}(\text{MH}(\mathbf{H}^{l-1}))),$$

$$\mathbf{H}^l = \text{LayerNorm}(\mathbf{G}^{l-1} + \text{Dropout}(\text{FFN}(\mathbf{G}^{l-1}))),$$

where $\mathbf{H}^l \in \mathbb{R}^{N \times d}$ is the hidden states at the l -th Transformer layer ($l = 1, \dots, L$). The weights $\mathbf{W}_Q^{(i)}, \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d \times (d/h)}$ are for the query, key, and value in attention head i , where h is the total number of heads. $\mathbf{W}_O \in \mathbb{R}^{d \times d}$ projects the multi-head attention output. We use $\mathbf{H}^L = [\mathbf{h}_1^L, \dots, \mathbf{h}_N^L]^\top$ (from the L -th Transformer layer) for next item prediction via a dot product with item embeddings.

Training. We use the binary cross-entropy loss on block D_t :

$$\mathcal{L}_{\text{BCE}} = - \sum_{S_u^t \in D_t} \sum_{j=1}^{N-1} \left[\log \sigma(\mathbf{e}_{i_{j+1}^t}^\top \mathbf{h}_j^L) + \sum_{i_{\text{neg}} \in \mathcal{I}_{\text{neg}}} \log(1 - \sigma(\mathbf{e}_{i_{\text{neg}}}^\top \mathbf{h}_j^L)) \right] \quad (2)$$

where σ is the sigmoid function, $\mathcal{I}_{\text{neg}} \subset \mathcal{I}^{1:t} \setminus S_u^t$ is the set of randomly sampled negative items per time step and $\mathcal{I}^{1:t} = \bigcup_{t'=1}^t \mathcal{I}^{t'}$ is the union of items seen along all previous data blocks D_1, \dots, D_t .

Inference. We compute relevance scores $\sigma(E_{\mathcal{I}^{1:t}} \mathbf{h}_N^L) \in [0, 1]^{|\mathcal{I}^{1:t}|}$ by taking the dot product between the item embeddings $E_{\mathcal{I}^{1:t}} \in \mathbb{R}^{|\mathcal{I}^{1:t}| \times d}$ and the hidden state at the last position $\mathbf{h}_N^L \in \mathbb{R}^d$, which

encodes information across all positions [35]. These scores are then used to rank items for next item prediction.

3.2.2 Linear Attention. Linear attention [5, 18, 40] was originally designed to approximate self-attention in linear time—reducing its quadratic $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. More recently, it has been extended to store past knowledge in parametric memories, thereby mitigating forgetting [22, 28]. Given the $(l-1)$ -th layer hidden states $\mathbf{H}^{l-1} \in \mathbb{R}^{N \times d}$, we form $\mathbf{Q} = \mathbf{H}^{l-1} \mathbf{W}_Q$, $\mathbf{K} = \mathbf{H}^{l-1} \mathbf{W}_K$, $\mathbf{V} = \mathbf{H}^{l-1} \mathbf{W}_V \in \mathbb{R}^{N \times d}$ (omitting the head-specific index for clarity) and denote their i -th rows by $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i \in \mathbb{R}^d$, respectively. A standard self-attention head computes $[\mathbf{a}_1, \dots, \mathbf{a}_N]^\top = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right) \mathbf{V} \in \mathbb{R}^{N \times d}$. In linear attention, the output at the i -th position $\mathbf{a}_i \in \mathbb{R}^d$ is computed:

$$\mathbf{a}_i = \frac{\sum_{j=1}^i \text{sim}(\mathbf{q}_i, \mathbf{k}_j) \mathbf{v}_j}{\sum_{j=1}^i \text{sim}(\mathbf{q}_i, \mathbf{k}_j)} \quad (3)$$

$\text{sim}(\mathbf{q}_i, \mathbf{k}_j)$ is expressed as $\phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j)$, using a kernel feature map function ϕ (e.g., ELU [6]). Subsequently, Eq.(3) is rewritten as:

$$\mathbf{a}_i = \frac{\sum_{j=1}^i \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j) \mathbf{v}_j}{\sum_{j=1}^i \phi(\mathbf{q}_i)^\top \phi(\mathbf{k}_j)} = \frac{\phi(\mathbf{q}_i)^\top \sum_{j=1}^i \phi(\mathbf{k}_j) \mathbf{v}_j}{\phi(\mathbf{q}_i)^\top \sum_{j=1}^i \phi(\mathbf{k}_j)} = \frac{\phi(\mathbf{q}_i)^\top \mathbf{s}_i}{\phi(\mathbf{q}_i)^\top \mathbf{z}_i}, \quad (4)$$

$$\text{where } \mathbf{s}_i = \sum_{j=1}^i \phi(\mathbf{k}_j) \mathbf{v}_j^\top, \quad \mathbf{z}_i = \sum_{j=1}^i \phi(\mathbf{k}_j) \quad (5)$$

Here, $\mathbf{s}_i \in \mathbb{R}^{d \times d}$ is called attention memory that encapsulates keys and values. Meanwhile, $\mathbf{z}_i \in \mathbb{R}^d$ is called normalizer memory that contains only keys. These memories are sequentially updated by combining the memories from the previous position (i.e., \mathbf{s}_{i-1} and \mathbf{z}_{i-1}) with the current values (i.e., $\phi(\mathbf{k}_i) \mathbf{v}_i^\top$ and $\phi(\mathbf{k}_i)$), similar to hidden state propagation in RNNs:

$$\mathbf{s}_i = \mathbf{s}_{i-1} + \phi(\mathbf{k}_i) \mathbf{v}_i^\top, \quad \mathbf{z}_i = \mathbf{z}_{i-1} + \phi(\mathbf{k}_i) \quad (6)$$

Note that \mathbf{s}_i and \mathbf{z}_i accumulate knowledge from position 1 to i in the input sequence S_u^t , serving as parametric memories [28]. Through these memories, linear attention partially retains historical knowledge without direct access to all previous positions [22], while approximating self-attention's expressiveness.

4 CSTRec

We present CSTRec, designed to effectively update Transformer-based SR model with data streams. We begin by discussing how linear attention can be applied naively to continual SR (§4.1). Next, we present Continual Sequential Attention, a specialized linear attention for continual SR (§4.2), and introduce a new technique to facilitate accommodation of new users (§4.3). Lastly, we present the optimization process (§4.4). Figure 1 shows an overview of CSTRec, with main notations summarized in Table ??.

4.1 Applying Linear Attention to Continual SR

We describe the operation of linear attention in continual SR through five steps: (1) retaining historical knowledge, (2) acquiring current knowledge, (3) integrating both knowledge, (4) computing attention, and (5) updating historical knowledge. Assume that the model has been trained up to data block D_{t-1} , and is now being updated with data block D_t . We omit the user index u for notational simplicity.

(1) **Retaining historical knowledge.** At this point, the model stores historical knowledge in two parametric memories per

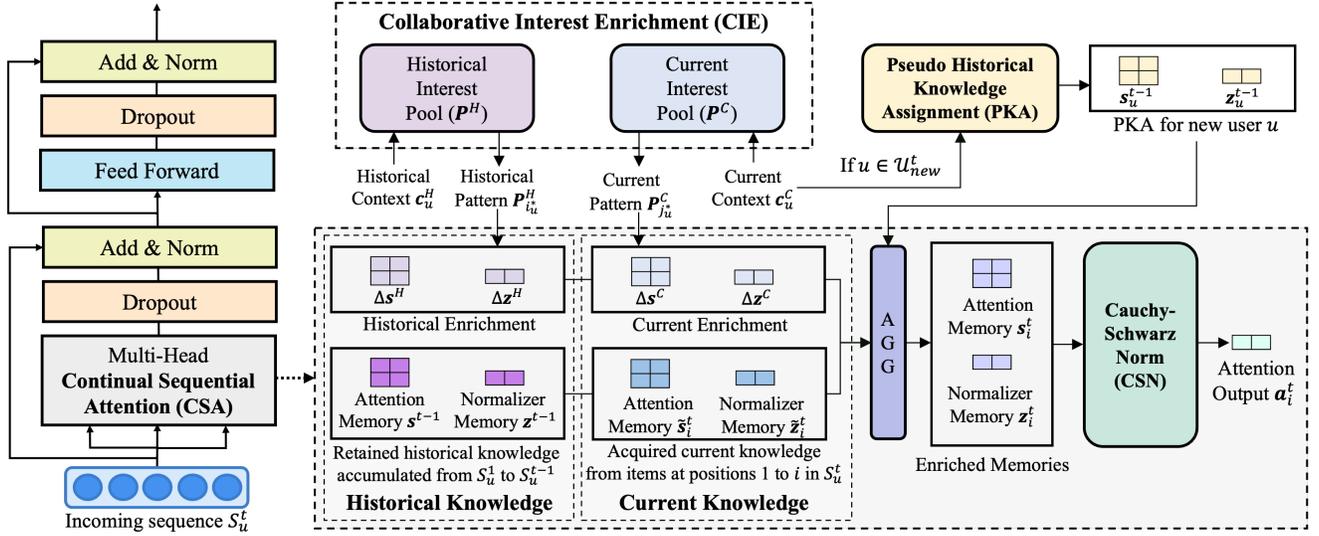


Figure 1: Overview of CSTRec, illustrating the computation of attention output \mathbf{a}_i^t for the i -th item in the incoming sequence S_u^t . Following the Multi-head CSA, we apply the same transformer sublayer architecture (§3.2.1)—dropout, layer normalization, and the position-wise feed-forward network. AGG denotes the aggregation of historical and current knowledge in Eq. (16).

user: attention memory \mathbf{s}^{t-1} and normalizer memory \mathbf{z}^{t-1} . The superscript denotes the time axis, indicating knowledge up to D_{t-1} . Note that these memories are continuously updated along the data stream (i.e., $\mathbf{s}^1 \rightarrow \dots \rightarrow \mathbf{s}^{t-2} \rightarrow \mathbf{s}^{t-1}$).

- (2) **Acquiring current knowledge.** When an incoming user sequence S_u^t arrives, the model encodes current knowledge into attention memory $\tilde{\mathbf{s}}_i^t$ and normalizer memory $\tilde{\mathbf{z}}_i^t$ as follows:

$$\tilde{\mathbf{s}}_i^t = \sum_{j=1}^i \phi(\mathbf{k}_j^t)(\mathbf{v}_j^t)^\top, \quad \tilde{\mathbf{z}}_i^t = \sum_{j=1}^i \phi(\mathbf{k}_j^t) \quad (7)$$

Note that these current memories (i.e., $\tilde{\mathbf{s}}_i^t$ and $\tilde{\mathbf{z}}_i^t$) capture only up-to-date knowledge from the first to the i -th item in S_u^t .

- (3) **Integrating historical and current knowledge.** We combine the attention and normalizer memories from both historical and current knowledge into \mathbf{s}_i^t and \mathbf{z}_i^t , respectively, as follows:

$$\mathbf{s}_i^t = \mathbf{s}^{t-1} + \tilde{\mathbf{s}}_i^t, \quad \mathbf{z}_i^t = \mathbf{z}^{t-1} + \tilde{\mathbf{z}}_i^t \quad (8)$$

- (4) **Computing attention output.** Finally, the attention output at the i -th position $\mathbf{a}_i^t \in \mathbb{R}^d$ is computed by rewriting Eq. (4):

$$\mathbf{a}_i^t = \frac{\phi(\mathbf{q}_i^t)^\top \mathbf{s}_i^t}{\phi(\mathbf{q}_i^t)^\top \mathbf{z}_i^t} \quad (9)$$

This attention output reflects the comprehensive context of both historical and current knowledge, allowing CSTRec to capture long-term user preferences along data streams.

- (5) **Updating historical knowledge.** After training on D_t , the model updates two parametric memories per user as follows:

$$\mathbf{s}^t = \mathbf{s}^{t-1} + \tilde{\mathbf{s}}_N^t, \quad \mathbf{z}^t = \mathbf{z}^{t-1} + \tilde{\mathbf{z}}_N^t, \quad (10)$$

where $\tilde{\mathbf{s}}_N^t = \sum_{j=1}^N \phi(\mathbf{k}_j^t)(\mathbf{v}_j^t)^\top$ and $\tilde{\mathbf{z}}_N^t = \sum_{j=1}^N \phi(\mathbf{k}_j^t)$ are the memories at the last position of the N -item sequence S_u^t . The updated memories \mathbf{s}^t and \mathbf{z}^t remain frozen during training.

However, this naive application yields suboptimal results due to two main challenges: First, **unstable learning** arises from an imbalance in the number of interactions per user along data streams. Because

the memories accumulate item representations from the given sequence over time (Eq. (10)), their magnitudes grow increasingly large for active users while remaining relatively small for less active users. This disproportion across users leads to wide variation in attention output magnitudes, with active users empirically producing larger values. This in turn makes optimization highly unstable (without CSN in Figure 2), and the instability worsens as more data accumulates over time. Second, **inevitable forgetting** occurs as new data continuously update the memories, gradually causing them to forget previously acquired knowledge. In the following sections, we introduce our solutions to these challenges.

4.2 Continual Sequential Attention (CSA)

We propose CSA, a tailored linear attention mechanism for continual SR. It comprises two key components: (1) Cauchy-Schwarz Normalization to prevent unstable learning, and (2) Collaborative Interest Enrichment to alleviate the inevitable forgetting.

4.2.1 Cauchy-Schwarz Normalization (CSN). We introduce CSN, a simple yet effective normalization technique based on the Cauchy-Schwarz inequality. CSN is applied during the computation of attention outputs to ensure stable learning. Let $\mathbf{q}_i^t = \phi(\mathbf{q}_i^t)$ and $\mathbf{a}_{\text{orig}} = \mathbf{a}_i^t$. Then, Eq. (9) becomes $\mathbf{a}_{\text{orig}} = \frac{(\mathbf{q}_i^t)^\top \mathbf{s}_i^t}{(\mathbf{q}_i^t)^\top \mathbf{z}_i^t}$. Here, we apply the Cauchy-Schwarz inequality to impose an upper bound on the denominator, thereby scaling the attention output magnitudes:

$$(\mathbf{q}_i^t)^\top \mathbf{z}_i^t \leq \|\mathbf{q}_i^t\|_2 \|\mathbf{z}_i^t\|_2 \quad (11)$$

By replacing the denominator in \mathbf{a}_{orig} with its Cauchy-Schwarz upper bound, we obtain \mathbf{a}_{csn} as follows:

$$\mathbf{a}_{\text{csn}} = \left(\frac{\mathbf{q}_i^t}{\|\mathbf{q}_i^t\|_2} \right)^\top \left(\frac{\mathbf{s}_i^t}{\|\mathbf{z}_i^t\|_2} \right) = \hat{\mathbf{q}}_i^\top \mathbf{r}_i^t, \quad (12)$$

where $\hat{\mathbf{q}}_i \in \mathbb{R}^d$ is the L2-normalized query, and $\mathbf{r}_i^t \in \mathbb{R}^{d \times d}$ is CSN attention memory, respectively. In terms of L2 norms, $\|\mathbf{a}_{\text{csn}}\|_2 =$

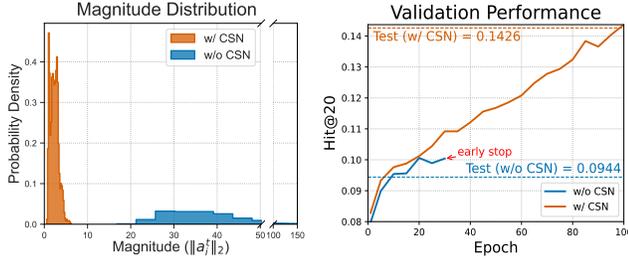


Figure 2: Effects of CSN on Yelp. (Left) Magnitude distribution of linear attention output. (Right) Validation performance. For ‘w/o CSN’, we apply layer normalization to the attention outputs to promote stable training.

$|\cos\theta_{qz}| \cdot \|a_{\text{orig}}\|_2$, where $\cos\theta_{qz} = \frac{(q_i^t)^T z_i^t}{\|q_i^t\| \|z_i^t\|} = \frac{a_{\text{CSN}}}{a_{\text{orig}}}$. As $0 \leq |\cos\theta_{qz}| \leq 1$, a_{CSN} adaptively adjusts the magnitude between 0 and the original scale, preventing excessive divergence and keeping them within a stable range. As a_{CSN} only modifies the overall magnitude, it maintains the information in a_{orig} . Figure 2 shows that CSN effectively scales the magnitudes, enabling stable and effective optimization while taking advantage of linear attention.

4.2.2 Collaborative Interest Enrichment (CIE). We propose CIE to alleviate the inevitable forgetting and enhance the learning process. As its core, CIE introduces two types of *interest pools*, each comprising a small set of learnable key–pattern pairs (i.e., $\mathcal{P} = \{(\kappa_i, \mathbf{P}_i)\}_{i=1}^M$), where each key $\kappa_i \in \mathbb{R}^d$ is used for matching, and each pattern $\mathbf{P}_i \in \mathbb{R}^{L \times d}$ encodes a distinct aspect of user interests. By leveraging this diverse interest knowledge, these pools are utilized to enrich the memories (i.e., s_i^t and z_i^t) by selecting the most relevant interest for each user context.

The motivations for introducing interest pools are twofold. First, unlike the backbone transformer parameters that are frequently updated through next-item prediction, each pool updates only the most relevant pattern for a given user context. This selective update preserves specific aspects of past knowledge and reduces overwriting. Second, the pools are globally shared across users, allowing similar users to collectively reinforce these patterns. As a result, they become more robust to forgetting and allow for leveraging recent behaviors of similar users when recalling historical patterns.

Historical and current interest pools. CIE leverages two interest pools: (1) Historical interest pool (\mathcal{P}^H), focusing on past interests to complement historical knowledge. (2) Current interest pool (\mathcal{P}^C), focusing on new interests to enrich current knowledge. Simply put, \mathcal{P}^H aims to preserve long-term preferences, while \mathcal{P}^C supplements sparse information on emerging user interests, thereby ensuring the model covers both historical and current knowledge.

We employ a key–pattern pair design [37, 44, 45], where relevant patterns are retrieved through a matching process. Each pool holds:

$$\begin{aligned} \mathcal{P}^H &= \{(\kappa_i^H, \mathbf{P}_i^H)\}_{i=1}^{N_H}, \quad \kappa_i^H \in \mathbb{R}^d, \quad \mathbf{P}_i^H \in \mathbb{R}^{L_H \times d}, \\ \mathcal{P}^C &= \{(\kappa_j^C, \mathbf{P}_j^C)\}_{j=1}^{N_C}, \quad \kappa_j^C \in \mathbb{R}^d, \quad \mathbf{P}_j^C \in \mathbb{R}^{L_C \times d} \end{aligned} \quad (13)$$

N and L are hyperparameters that determine each pool’s capacity: N specifies the number of interests stored (variety of knowledge), and L specifies the length of each interest (level of detail). We provide a detailed study in §5.3.3.

Interest enrichment. Given an input sequence $S_u^t = [i_1^t, \dots, i_N^t]$, we obtain historical and current contexts $\mathbf{c}_u^H, \mathbf{c}_u^C \in \mathbb{R}^d$ from its last hidden state from the final Transformer layer.² Using these contexts, we retrieve the most relevant interest in each pool. For \mathbf{c}_u^H and \mathbf{c}_u^C , we identify the best-matching indices i_u^* and j_u^* in pools \mathcal{P}^H and \mathcal{P}^C , respectively, using a matching function γ .³

$$i_u^* = \underset{i \in \{1, \dots, N_H\}}{\operatorname{argmin}} \gamma(\mathbf{c}_u^H, \kappa_i^H), \quad j_u^* = \underset{j \in \{1, \dots, N_C\}}{\operatorname{argmin}} \gamma(\mathbf{c}_u^C, \kappa_j^C) \quad (14)$$

Note that we store the best-matching indices and use them at the inference phase, enabling interest enrichment with negligible costs. The corresponding interest knowledge $\mathbf{P}_{i_u^*}^H$ and $\mathbf{P}_{j_u^*}^C$ form historical interest enrichment ($\Delta s^H, \Delta z^H$) and current interest enrichment ($\Delta s^C, \Delta z^C$), respectively. These are computed using the key- and value-projection matrices \mathbf{W}_K and \mathbf{W}_V , as in Eq. (5):

$$\begin{aligned} \Delta s^H &= \sum_{l=1}^{L_H} \phi(\{\mathbf{P}_{i_u^*}^H[l, :] \mathbf{W}_K\} (\mathbf{P}_{i_u^*}^H[l, :] \mathbf{W}_V)^\top), \quad \Delta z^H = \sum_{l=1}^{L_H} \phi(\mathbf{P}_{i_u^*}^H[l, :] \mathbf{W}_K) \\ \Delta s^C &= \sum_{l=1}^{L_C} \phi(\{\mathbf{P}_{j_u^*}^C[l, :] \mathbf{W}_K\} (\mathbf{P}_{j_u^*}^C[l, :] \mathbf{W}_V)^\top), \quad \Delta z^C = \sum_{l=1}^{L_C} \phi(\mathbf{P}_{j_u^*}^C[l, :] \mathbf{W}_K) \end{aligned} \quad (15)$$

We then extend Eq. (8) to aggregate both aspects as follows:

$$\begin{aligned} s_i^t &= \underbrace{s_i^{t-1} + \Delta s^H}_{\text{Historical knowledge}} + \underbrace{\tilde{s}_i^t + \Delta s^C}_{\text{Current knowledge}} \\ z_i^t &= \underbrace{z_i^{t-1} + \Delta z^H}_{\text{Historical knowledge}} + \underbrace{\tilde{z}_i^t + \Delta z^C}_{\text{Current knowledge}} \end{aligned} \quad (16)$$

The enriched memories s_i^t and z_i^t from Eq. (16) are then passed to the CSN step in Eq. (12). Lastly, we introduce the following loss for accurate matching:

$$\mathcal{L}_{\text{match}} = \frac{1}{|D_t|} \sum_{S_u^t \in D_t} \left[\gamma(\mathbf{c}_u^H, \kappa_{i_u^*}^H) + \gamma(\mathbf{c}_u^C, \kappa_{j_u^*}^C) \right], \quad (17)$$

This loss function pulls selected keys closer to corresponding contexts, allowing for progressively capturing more accurate interests. As a result, historical and current interests encode distinguishable knowledge. Please refer to §5.3.4 for further analysis.

4.3 Pseudo-Historical Knowledge Assignment

One critical challenge in continual SR is accommodating newly joined users who have no past interactions. At time period t , new users have current memories $(\tilde{s}_i^t, \tilde{z}_i^t)$, but has no historical ones (s_i^{t-1}, z_i^{t-1}) . To address this, we introduce a pseudo-historical knowledge assignment that leverages the historical knowledge of existing users with the most similar behavioral patterns. For each new user, the process involves two steps: (1) identifying the top- K existing users with similar current interests, and (2) assigning pseudo-historical knowledge derived from their historical knowledge.

First, for each new user u , we identify top- K existing users \mathcal{N}_u :

$$\mathcal{N}_u = \{u'\} \underset{u' \in \mathcal{U}^t \setminus \mathcal{U}_{\text{new}}^t}{\operatorname{argsort}} \gamma(\mathbf{c}_u^C, \mathbf{c}_{u'}^C)[:K], \quad (18)$$

²The computation follows the Transformer layer with CSA (§3.2.1). During the attention computation in Eq. (12), we use $(\tilde{s}_i^t, \tilde{z}_i^t)$ for \mathbf{c}_u^C to better focus on the current interest, while leveraging (s_i^t, z_i^t) to reflect both historical and current aspects for \mathbf{c}_u^H .
³In this work, we use cosine distance for its simplicity.

Algorithm 1: CSTRec algorithm on t -th data block (D_t)

Input : Data block $D_t = \{S_u^t\}_{u \in \mathcal{U}^t}$, Model $M(\cdot; \theta)$, Pools $\mathcal{P}^H, \mathcal{P}^C$, Memories $\{s_u^{t-1}, z_u^{t-1} \mid u \in \mathcal{U}^t \setminus \mathcal{U}_{\text{new}}^t\}$

Output : Updated model $M(\cdot; \theta)$, Updated Pools $\mathcal{P}^H, \mathcal{P}^C$, Updated Memories $\{s_u^t, z_u^t \mid u \in \mathcal{U}^t\}$

- 1 for each epoch do
- 2 if epoch % $R = 0$ then
- 3 Assign pseudo-historical knowledge $s_u^{t-1}, z_u^{t-1}, \forall u \in \mathcal{U}_{\text{new}}^t$ ▶ Eq. (19)
- 4 Identify the best-matching indices $i_u^*, j_u^*, \forall u \in \mathcal{U}^t$ ▶ Eq. (14)
- 5 for each CSTRec layer do
- 6 for each head do
- 7 Acquire current memories at the i -th position $\tilde{s}_i^t, \tilde{z}_i^t$ ▶ Eq. (7)
- 8 Retrieve historical memories s^{t-1}, z^{t-1}
- 9 Derive $(\Delta s^H, \Delta z^H, \Delta s^C, \Delta z^C)$ then enrich (s_i^t, z_i^t) ▶ Eq. (15), (16)
- 10 Compute attention output a_i^t using CSN ▶ Eq. (12)
- 11 Aggregate multi-head results, followed by FFN, Dropout, LayerNorm
- 12 Optimize the parameters by minimizing the loss ▶ Eq.(20)
- 13 Update historical memories $s_u^t, z_u^t, \forall u \in \mathcal{U}^t$ ▶ Eq. (10)

where $\mathcal{U}^t \setminus \mathcal{U}_{\text{new}}^t$ is the set of existing users in D_t , excluding new users. Here, c_u^t is the context of current interest, obtained from the last hidden state of S_u^t (§4.2.2). Next, we generate pseudo-historical knowledge, weighted by their similarity scores $\psi(u, u')$ as:

$$s_u^{t-1} = \sum_{u' \in \mathcal{N}_u} \psi(u, u') s_{u'}^{t-1}, \quad z_u^{t-1} = \sum_{u' \in \mathcal{N}_u} \psi(u, u') z_{u'}^{t-1}, \quad (19)$$

where s_u^{t-1} and z_u^{t-1} are the pseudo-historical knowledge for the new user u . We use the Softmax function to obtain normalized weight.⁴ Now, s_u^{t-1} and z_u^{t-1} serve as historical knowledge for CSA computation. This technique complements the lack of historical data for new users, facilitating their adaptation in CSTRec.

4.4 Optimization of CSTRec

The final learning objective of CSTRec is as follows:

$$\min_{\theta, \mathcal{P}^H, \mathcal{P}^C} \mathcal{L}_{\text{BCE}} + \lambda_{\text{match}} \mathcal{L}_{\text{match}} \quad (20)$$

where $\theta, \mathcal{P}^H, \mathcal{P}^C$ are the parameters of CSTRec and interest pools. \mathcal{L}_{BCE} and $\mathcal{L}_{\text{match}}$ are defined in Eqs. (2) and (17), and λ_{match} is a hyperparameter for balancing the losses. The training process is detailed in Algorithm 1. Pseudo knowledge assignment (line 3) and interest pool retrieval (line 4) are performed every R epochs, as conducting them every epoch is unnecessary and time-consuming.

Time Complexity Analysis. For each transformer layer in CSTRec, the CSA head with the CIE module requires $\mathcal{O}(Nd^2)$ for linear attention (Eq. (4)) and $\mathcal{O}((L_H + L_C)d^2)$ for interest enrichment (Eq. (15)), yielding $\mathcal{O}((N + L_H + L_C)d^2) \approx \mathcal{O}(Nd^2)$. This further simplifies to $\mathcal{O}(N)$, since $N \gg d$ and d is a fixed constant [26, 49]. Identifying the best-matching indices (Eq. (14)) and top- K existing users (Eq. (18)) requires $\mathcal{O}(|\mathcal{U}^t|(N_H + N_C)d)$ and $\mathcal{O}(|\mathcal{U}_{\text{new}}^t|(|\mathcal{U}^t| - |\mathcal{U}_{\text{new}}^t|)d)$, respectively. Both operations run only once every R epochs—not per layer or per position—so their amortized overhead per epoch is negligible compared to the cost of a single transformer layer.

Efficiency of CSTRec. Compared to the self-attention, CSTRec shows comparable efficiency for training and greatly reduced efficiency for inference via three key designs: (1) Building upon linear attention, CSTRec reduces computational complexity with respect to the input length from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. (2) During training,

⁴ $\psi(u, u') = \exp(c_u^C \cdot c_{u'}^C / \tau) / \sum_{u' \in \mathcal{N}_u} \exp(c_u^C \cdot c_{u'}^C / \tau)$

Table 1: Data block statistics after preprocessing.

Data Blocks		D ₀ (60%)	D ₁ (10%)	D ₂ (10%)	D ₃ (10%)	D ₄ (10%)
Gowalla	# of users (new users)	30,682(30,682)	2,364(692)	2,227(828)	2,334(902)	2,490(1,082)
	# of items (new items)	68,189(68,189)	3,006(920)	2,879(1,059)	3,000(1,123)	3,076(1,169)
	# of interactions	1,754,145	49,637	45,738	46,956	49,127
	Avg. Seq Length	57.17	21.00	20.54	20.12	19.73
ML-1M	Sparsity	0.9992	0.9930	0.9929	0.9933	0.9936
	# of users (new users)	3,978(3,978)	820(777)	767(567)	844(597)	932(109)
	# of items (new items)	2,845(2,845)	1,680(6)	1,768(4)	1,726(0)	1,876(18)
	# of interactions	498,877	77,470	77,382	76,933	76,619
Yelp	Avg. Seq Length	125.41	94.48	100.89	91.15	82.21
	Sparsity	0.9559	0.9438	0.9429	0.9472	0.9562
	# of users (new users)	104,281(104,281)	7,340(3,634)	7,065(3,820)	7,173(3,643)	8,260(4,312)
	# of items (new items)	52,290(52,290)	5,736(669)	5,708(843)	6,283(1,093)	7,183(1,401)
Yelp	# of interactions	1,449,055	66,365	64,364	68,152	84,820
	Avg. Seq Length	13.89	9.04	9.11	9.50	10.26
	Sparsity	0.9997	0.9984	0.9984	0.9985	0.9986

CSTRec efficiently performs CIE and pseudo-historical knowledge assignment at predefined intervals R , reducing overhead. (3) During inference, CSTRec leverages the best-matching indices identified during the training, enabling efficient CIE with negligible costs. A detailed analysis of efficiency is provided in Table 4.

5 Experiments

5.1 Experimental Setup

5.1.1 Datasets. We use three real-world datasets: Gowalla, ML-1M, and Yelp, following [21, 26, 42, 49]. To simulate non-stationary data streams, we split each dataset chronologically. The first 60% of the data serves as the base block (D_0), which is used to pretrain all methods before the continual updates begin. The remaining 40% is equally divided into four incremental blocks (D_1 to D_4), following prior CL studies [21, 27, 50]. We apply k -core filtering with $k = 5$ for Yelp and $k = 10$ for other datasets for each block. Each interaction sequence in a block is split into training, validation, and test sets based on item positions: the last item serves as the test label, the second-to-last item as the validation label, and the remaining items are used for training. Table 1 presents detailed statistics.

5.1.2 Evaluation metrics. Recent CL studies [9, 21] have employed specialized metrics to evaluate a model’s ability to retain and acquire knowledge over time. Following these work, we adopt two CL metrics: *Retained Average (RA)* and *Learning Average (LA)*. Specifically, we construct a performance matrix $A \in \mathbb{R}^{t \times t}$, where each entry a_{ij} (with $i \geq j$) represents the recommendation performance on block j after training on block i . After updating the model on the t -th block (i.e., D_t), we report the following metrics:

- **RA** : $\frac{1}{t} \sum_{i=1}^t a_{t,i}$ evaluates knowledge retention from past blocks.
 - **LA** : $\frac{1}{t} \sum_{j=1}^t a_{t,j}$ evaluates knowledge acquisition from new blocks. Additionally, we report *H-mean*, the harmonic mean of LA and RA, to provide an overall comparison of a model’s capability [9]. We use Hit@20 (H@20), MRR@20 (M@20) [39], and NDCG@20 (N@20) [15] as evaluation metrics [9, 42, 50]. All results are averaged over five independent runs with different random seeds. Note that we report results after training on D_2, D_3 and D_4 in Tables 2 and 3, as forgetting becomes evident at those stages, but not on D_1 .
- 5.1.3 Baselines.** For a thorough evaluation, we adopt two distinct training setups and compare state-of-the-art methods for each setup.
- (1) **Fine-tune:** The model is continually updated using only the incoming data block, without direct access to historical data. We compare the following state-of-the-art CL methods:
 - **SAIL-PIW** [42] is a regularization-based method that assigns regularization weights based on user preference shift.

- **ReLoop2** [59] is a replay-based method that uses an error memory module to improve future recommendations.
 - **IMSR** [43] is an incremental learning framework for SR, which utilizes multiple interest representations for each user.
- (2) **Full-batch**: The model is updated with all incremental blocks. We compare recent methods designed to capture user interests formed over long periods (i.e., lifelong or long-term SR models):
- **HPMN** [31] uses GRUs for personalized memorization to capture multi-scale sequential patterns in lifelong sequences.
 - **LimaRec** [49] builds upon a linear attention mechanism to capture multi interests of users within lifelong sequences.
 - **LinRec** [26] proposes an L2-normalized linear attention mechanism that leverages dual-side normalization techniques.

As a representative transformer-based SR model, we use SASRec [17] for performance comparison in both setups. For baselines without their own architecture (i.e., SAIL-PIW and ReLoop2), we integrate their modules into SASRec to evaluate their performance on SR.

5.1.4 Implementation details. We utilize PyTorch with CUDA, utilizing RTX 3090 GPU and AMD EPYC 7413 CPU. Hyperparameters are tuned through grid search on the validation set. The learning rate is chosen from $\{0.0001, 0.0002, 0.0005\}$. L_2 regularization for Adam is chosen from $\{5e-6, 1e-5, 5e-5, 1e-4\}$, and the dropout ratio from $\{0.05, 0.075, 0.1, 0.125\}$. The number of heads h , layers L , and negative samples are set to 2. The dimension d is set to 64 for ML-1M, 32 for Gowalla, and 16 for Yelp. In the fine-tune setup, the window sizes are set to 50, 25, and 10 for each dataset, respectively, and are doubled in the full-batch setup to better capture long-term dependencies. These choices align with the guideline of maintaining a ratio of sequence length to dimension greater than 1.5 for long-term SR scenarios [26], while also considering the average sequence lengths. For CSTRec, the number of interests N_H, N_C is chosen from $\{10, 20, \dots, 50\}$, interest lengths L_H, L_C from $\{10, 20, 50\}$. The number of similar users K for pseudo-historical knowledge assignment is chosen from $\{5, 10, 15, 20, 25\}$. We fix $\lambda_{\text{match}} = 1e-4$ (performance is largely insensitive to this choice), $\tau = 1.0$, and $C = 5$. For baseline-specific hyperparameters, we follow the search ranges reported in the original papers.

5.2 Performance Comparison

Table 2 and Table 3 show the overall performance under the fine-tune and full-batch setups, respectively. In both setups, CSTRec consistently outperforms the baselines by effectively retaining historical knowledge (RA) while acquiring current knowledge (LA), achieving a better balance between them (H-mean).

5.2.1 Fine-tune setup. Overall, CSTRec shows superior performance across all data blocks compared to state-of-the-art CL methods, including regularization-based (i.e., SAIL-PIW), replay-based (i.e., ReLoop2), and multi-interest incremental SR (i.e., IMSR) approaches. Unlike conventional CL methods that gradually dilute historical knowledge, CSTRec, equipped with CSA (featuring CSN and CIE), effectively preserves historical user interests (reflected in improved RA) while leveraging them to facilitate the learning of current user interests (reflected in LA). Moreover, CSTRec strategically leverages existing users’ historical knowledge to enhance adaptation for new users, further improving overall performance.

Analysis on various user groups. For a more thorough assessment of the model’s capabilities in (1) knowledge retention, (2) knowledge acquisition, and (3) balancing these aspects, we perform a user-level analysis. We compare two CL baselines, IMSR and SAIL-PIW, which show competitive results in the main tables. After fine-tuning on D_4 , we report the results on three distinct user groups:

- (1) **Dormant users** who interact only in D_1 and D_4 (i.e., inactive during D_2 and D_3).
- (2) **New users** who are newly joined in D_4 .
- (3) **Active users** who interact across all blocks, from D_1 to D_4 .

Figure 3 shows that CSTRec consistently outperforms IMSR and SAIL-PIW for all user groups. These results collectively support the superiority of CSTRec in retaining historical knowledge (dormant users), adapting to current knowledge (new users), and balancing them to provide high-quality recommendations (active users).

5.2.2 Full-batch setup. Compared to various long-term SR baselines, including self-attention (i.e., SASRec), and RNN-based (i.e., HPMN), and linear attention (i.e., LimaRec and LinRec) methods, CSTRec more effectively captures knowledge from data streams, as evidenced by improved performance across data blocks. In general, self- and linear attention approaches outperform the RNN-based method by capturing long-range dependencies. CSTRec further enhances performance by capturing the trajectory of user interests over time. Specifically, CIE provides additional user-specific guidance by leveraging collaborative signals to enrich both historical and current user interests. Moreover, CSN ensures stable learning along data streams, addressing the instability often observed in training linear attention-based methods. This complementary synergy is reflected in the enhanced H -mean performance.

In summary, CSTRec enables effective adaptation to continuously arriving data while mitigating the forgetting of previously acquired knowledge during fine-tuning. CSTRec also shows strong capabilities in capturing long-term preferences from data streams and improving recommendation quality in full-batch setup. These results collectively support the effectiveness of CSTRec in handling continuously incoming user behavior sequences for continual SR.

5.3 Study of CSTRec

We provide comprehensive analyses of CSTRec. In this section, we report the results on Gowalla dataset.

5.3.1 Accuracy and efficiency analysis. We compare the accuracy and efficiency of three attention mechanisms: self-attention (SASRec), linear attention (LimaRec, LinRec), and CSA (CSTRec). For CSA, we also compare ‘CSN only’, which excludes all other components, to verify CSN’s standalone efficacy. Table 4 presents the results on D_4 under a full-batch setup. Here, training time indicates the total time to complete the training process, and inference time measures the time to generate recommendations for all users. **Accuracy aspect.** CSA (CSN only) shows significant performance gains over linear attention (LimaRec), highlighting the effectiveness of CSN for stable optimization of linear attention in the continual SR problem. Moreover, CSA (CSTRec) further improves accuracy through CIE, which enriches historical and current interests in a complementary manner, as also evidenced by its superior performance over self-attention (SASRec) and linear attention (LinRec).

Table 2: Overall performance comparison for fine-tune setup. * indicates $p < 0.05$ for the paired t-test against the best baseline.

Fine-tune		RA	After D_2 LA	H-mean	RA	After D_3 LA	H-mean	RA	After D_4 LA	H-mean	
Gowalla	Hit@20	SASRec	0.5157	0.6242	0.5648	0.4456	0.6423	0.5261	0.3890	0.6486	0.4863
		SAIL-PIW	0.5153	0.6239	0.5644	0.4476	0.6426	0.5276	0.3901	0.6492	0.4874
		Reloop2	0.5158	0.6243	0.5649	0.4456	0.6423	0.5261	0.3890	0.6486	0.4863
		IMSR	0.5177	0.6162	0.5627	0.4526	0.6350	0.5285	0.3980	0.6412	0.4912
	CSTRec	0.5356*	0.6281*	0.5782*	0.4747*	0.6468*	0.5476*	0.4247*	0.6545*	0.5151*	
	MRR@20	SASRec	0.3806	0.4661	0.4191	0.3128	0.4724	0.3763	0.2550	0.4728	0.3313
		SAIL-PIW	0.3814	0.4632	0.4192	0.3150	0.4725	0.3780	0.2553	0.4735	0.3318
		Reloop2	0.3806	0.4661	0.4191	0.3127	0.4724	0.3763	0.2548	0.4728	0.3312
		IMSR	0.3828	0.4498	0.4136	0.3187	0.4596	0.3764	0.2609	0.4606	0.3332
	CSTRec	0.3939*	0.4631	0.4257*	0.3301*	0.4731	0.3889*	0.2681*	0.4747	0.3423*	
	NDCG@20	SASRec	0.4118	0.5029	0.4329	0.3433	0.5120	0.4110	0.2854	0.5139	0.3670
		SAIL-PIW	0.4123	0.5022	0.4328	0.3455	0.5121	0.4126	0.2860	0.5145	0.3677
Reloop2		0.4117	0.5029	0.4328	0.3433	0.5119	0.4110	0.2853	0.5138	0.3669	
IMSR		0.4140	0.4885	0.4482	0.3494	0.5002	0.4114	0.2920	0.5026	0.3693	
CSTRec	0.4265*	0.5015	0.4610*	0.3631*	0.5135	0.4254*	0.3035*	0.5164	0.3820*		
ML-1M	Hit@20	SASRec	0.5783	0.7710	0.6609	0.4768	0.7594	0.5858	0.3511	0.7119	0.4703
		SAIL-PIW	0.5807	0.7726	0.6630	0.4780	0.7608	0.5872	0.3525	0.7136	0.4719
		Reloop2	0.5785	0.7714	0.6612	0.4771	0.7597	0.5861	0.3513	0.7122	0.4706
		IMSR	0.5754	0.7646	0.6566	0.4763	0.7479	0.5820	0.3484	0.6969	0.4646
	CSTRec	0.5857*	0.7815*	0.6696*	0.4830*	0.7714*	0.5940*	0.3555	0.7311*	0.4784*	
	MRR@20	SASRec	0.1383	0.1755	0.1547	0.1139	0.1699	0.1363	0.0780	0.1535	0.1034
		SAIL-PIW	0.1387	0.1764	0.1553	0.1143	0.1708	0.1369	0.0785	0.1544	0.1041
		Reloop2	0.1384	0.1755	0.1548	0.1136	0.1699	0.1362	0.0781	0.1533	0.1034
		IMSR	0.1385	0.1756	0.1549	0.1149	0.1690	0.1368	0.0778	0.1517	0.1029
	CSTRec	0.1484*	0.1940*	0.1682*	0.1204*	0.1885*	0.1470*	0.0823*	0.1732*	0.1115*	
	NDCG@20	SASRec	0.2334	0.3041	0.2641	0.1921	0.2971	0.2333	0.1364	0.2735	0.1821
		SAIL-PIW	0.2342	0.3051	0.2650	0.1928	0.2980	0.2341	0.1371	0.2746	0.1829
Reloop2		0.2334	0.3042	0.2642	0.1920	0.2971	0.2332	0.1365	0.2733	0.1821	
IMSR		0.2329	0.3028	0.2633	0.1929	0.2938	0.2328	0.1357	0.2688	0.1804	
CSTRec	0.2434*	0.3218*	0.2772*	0.1990*	0.3152*	0.2440*	0.1410*	0.2940*	0.1906*		
Yelp	Hit@20	SASRec	0.1076	0.1220	0.1143	0.0985	0.1242	0.1098	0.0886	0.1263	0.1041
		SAIL-PIW	0.1108	0.1249	0.1174	0.1001	0.1320	0.1138	0.0873	0.1349	0.1059
		Reloop2	0.1076	0.1219	0.1143	0.0985	0.1243	0.1099	0.0886	0.1263	0.1042
		IMSR	0.1075	0.1206	0.1137	0.0973	0.1232	0.1087	0.0890	0.1253	0.1042
	CSTRec	0.1175*	0.1383*	0.1271*	0.1048*	0.1427*	0.1208*	0.0895	0.1431*	0.1101*	
	MRR@20	SASRec	0.0232	0.0251	0.0241	0.0210	0.0257	0.0231	0.0189	0.0260	0.0219
		SAIL-PIW	0.0243	0.0260	0.0251	0.0211	0.0271	0.0237	0.0185	0.0278	0.0222
		Reloop2	0.0232	0.0251	0.0241	0.0210	0.0257	0.0231	0.0189	0.0260	0.0219
		IMSR	0.0236	0.0253	0.0244	0.0208	0.0258	0.0230	0.0187	0.0260	0.0217
	CSTRec	0.0250	0.0282*	0.0265*	0.0216	0.0277*	0.0246*	0.0190	0.0287	0.0226	
	NDCG@20	SASRec	0.0411	0.0457	0.0433	0.0376	0.0466	0.0415	0.0337	0.0473	0.0394
		SAIL-PIW	0.0426	0.0470	0.0448	0.0378	0.0494	0.0428	0.0331	0.0505	0.0400
Reloop2		0.0411	0.0457	0.0433	0.0375	0.0466	0.0415	0.0337	0.0473	0.0394	
IMSR		0.0414	0.0455	0.0434	0.0371	0.0464	0.0412	0.0336	0.0471	0.0392	
CSTRec	0.0446*	0.0515*	0.0479*	0.0393*	0.0528*	0.0451*	0.0338	0.0529*	0.0412*		

Table 3: Overall performance comparison for full-batch setup. * indicates $p < 0.05$ for the paired t-test against the best baseline.

Full-batch		RA	After D_2 LA	H-mean	RA	After D_3 LA	H-mean	RA	After D_4 LA	H-mean	
Gowalla	Hit@20	SASRec	0.7125	0.7117	0.7121	0.7190	0.7172	0.7181	0.7155	0.7137	0.7146
		HPMN	0.6916	0.6886	0.6901	0.6991	0.6939	0.6965	0.6973	0.6914	0.6943
		LimaRec	0.6537	0.6286	0.6409	0.6743	0.6333	0.6332	0.6848	0.6354	0.6592
		LinRec	0.7132	0.7117	0.7125	0.7169	0.7159	0.7164	0.7138	0.7128	0.7133
	CSTRec	0.7200*	0.7177*	0.7189*	0.7263*	0.7230*	0.7246*	0.7218*	0.7188*	0.7203*	
	MRR@20	SASRec	0.5906	0.5732	0.5818	0.5938	0.5703	0.5813	0.5850	0.5690	0.5769
		HPMN	0.5846	0.5745	0.5795	0.5673	0.5587	0.5629	0.5625	0.5554	0.5590
		LimaRec	0.4788	0.3978	0.4338	0.4714	0.3846	0.4231	0.5062	0.3949	0.4434
		LinRec	0.6080	0.5881	0.5979	0.6026	0.5832	0.5927	0.5819	0.5759	0.5789
	CSTRec	0.6077	0.5920	0.5997	0.6054*	0.5867	0.5958*	0.5834	0.5731	0.5782	
	NDCG@20	SASRec	0.6190	0.6056	0.6122	0.6231	0.6048	0.6138	0.6156	0.6029	0.6092
		HPMN	0.6095	0.6011	0.6053	0.5981	0.5904	0.5942	0.5940	0.5873	0.5906
LimaRec		0.5191	0.4505	0.4820	0.5187	0.4416	0.4767	0.5479	0.4501	0.4941	
LinRec		0.6326	0.6170	0.6247	0.6292	0.6143	0.6217	0.6119	0.6072	0.6096	
CSTRec	0.6339*	0.6214	0.6276	0.6335*	0.6186*	0.6259*	0.6158*	0.6073	0.6115		
ML-1M	Hit@20	SASRec	0.6669	0.7208	0.6928	0.5598	0.6485	0.6009	0.4559	0.5676	0.5057
		HPMN	0.1321	0.1387	0.1353	0.1229	0.1301	0.1264	0.1082	0.1184	0.1131
		LimaRec	0.4554	0.4919	0.4728	0.3959	0.4418	0.4174	0.3411	0.3859	0.3619
		LinRec	0.4783	0.5314	0.5035	0.4168	0.4807	0.4465	0.3446	0.4194	0.3783
	CSTRec	0.6764*	0.7248	0.6998*	0.5799*	0.6548*	0.6151*	0.4785*	0.5776*	0.5234*	
	MRR@20	SASRec	0.1542	0.1738	0.1635	0.1295	0.1536	0.1405	0.1063	0.1321	0.1178
		HPMN	0.0229	0.0239	0.0234	0.0219	0.0227	0.0223	0.0197	0.0210	0.0203
		LimaRec	0.1084	0.1160	0.1120	0.0966	0.1050	0.1005	0.0830	0.0912	0.0869
		LinRec	0.1206	0.1336	0.1268	0.1052	0.1201	0.1122	0.0865	0.1040	0.0945
	CSTRec	0.1658*	0.1901*	0.1771*	0.1381*	0.1660*	0.1508*	0.1123*	0.1430*	0.1258*	
	NDCG@20	SASRec	0.2646	0.2925	0.2779	0.2216	0.2604	0.2395	0.1811	0.2359	0.2010
		HPMN	0.0458	0.0479	0.0468	0.0431	0.0452	0.0441	0.0383	0.0414	0.0398
LimaRec		0.1828	0.1968	0.1895	0.1609	0.1774	0.1686	0.1384	0.1544	0.1459	
LinRec		0.1979	0.2196	0.2082	0.1724	0.1980	0.1843	0.1421	0.1720	0.1557	
CSTRec	0.2764*	0.3069*	0.2909*	0.2332*	0.2722*	0.2512*	0.1909*	0.2371*	0.2115*		
Yelp	Hit@20	SASRec	0.1012	0.0981	0.0997	0.1098	0.0986	0.1039	0.1144	0.1012	0.1074
		HPMN	0.1537	0.1462	0.1499	0.1438	0.1436	0.1437	0.1299	0.1389	0.1343
		LimaRec	0.1018	0.1010	0.1014	0.0928	0.0930	0.0929	0.0864	0.0868	0.0865
		LinRec	0.1440	0.1299	0.1366	0.1388	0.1286	0.1335	0.1289	0.1259	0.1274
	CSTRec	0.1647*	0.1605*	0.1626*	0.1486*	0.1529*	0.1507*	0.1290	0.1425*	0.1353	
	MRR@20	SASRec	0.0221	0.0214	0.0218	0.0238	0.0210	0.0223	0.0249	0.0218	0.0233
		HPMN	0.0321	0.0301	0.0311	0.0300	0.0294	0.0297	0.0274	0.0284	0.0279
		LimaRec	0.0220	0.0217	0.0219	0.0203	0.0199	0.0201	0.0185	0.0185	0.0185
		LinRec	0.0311	0.0278	0.0294	0.0295	0.0272	0.0283	0.0275	0.0267	0.0271
	CSTRec	0.0343*	0.0325*	0.0334*	0.0308*	0.0307*	0.0307*	0.0279	0.0289	0.0284	
	NDCG@20	SASRec	0.0390	0.0378	0.0384	0.0420	0.0376	0.0397	0.0439	0.0387	0.0411
		HPMN	0.0580	0.0547	0.0563	0.0541	0.0536	0.0539	0.0492	0.0518	0.0505
LimaRec		0.0390	0.0385	0.038							

Table 4: Accuracy and efficiency of attention mechanisms.

Method	H@20	M@20	N@20	Training time (s)	Inference time (s)
Self-attention (SASRec)	0.7162	0.5828	0.6142	225.26	4.57
Linear attention (LimaRec)	0.6730	0.4754	0.5214	249.43	3.46
Linear attention (LinRec)	0.7102	0.5847	0.6143	213.90	2.74
CSA (CSN only)	0.7168	0.5888	0.6188	177.70	1.97
CSA (CSTRec)	0.7218	0.5965	0.6259	216.12	2.44

Table 5: Ablation study on Gowalla under the fine-tune setup.

CSN	CIE-H	CIE-C	PKA	RA	LA	H-mean	H-mean Imp (%)
x	x	x	x	0.3847	0.6307	0.4779	0.00
x	o	o	o	0.3908	0.6321	0.4830	+1.07
o	x	o	o	0.4142	0.6550	0.5075	+6.19
o	o	x	o	0.4166	0.6533	0.5088	+6.47
o	o	o	x	0.4074	0.6517	0.5014	+4.50
o	o	o	o	0.4247	0.6545	0.5151	+7.78

Table 6: New-user performance with top- K users in PKA.

K	0 (no PKA)	5	10	15	20	25	Imp (%)
H@20	0.6677	0.6731	0.6711	0.6714	0.6731	0.6741	+0.96
M@20	0.4181	0.4231	0.4214	0.4192	0.4193	0.4305	+2.97
N@20	0.4766	0.4818	0.4799	0.4786	0.4786	0.4877	+2.33

Efficiency aspect. Compared to the self-attention, CSA (CSTRec) shows comparable efficiency for training and greatly reduced efficiency for inference. Building upon the linear attention, CSA maintains the linear complexity with respect to the input length. Also, the additional computation introduced by the proposed modules is negligible compared to the baseline attention mechanisms. In contrast, self-attention (SASRec) incurs the highest inference time due to its quadratic complexity. Linear attention (LinRec) incurs relatively high inference costs due to the repeated application of its dual-side normalization techniques, while Linear attention (LimaRec) shows the slowest training time because its multi-interest module for all users adds extra computational overhead.

5.3.2 Ablation study. Table 5 presents a detailed ablation study on D_4 of Gowalla. CIE-H and CIE-C denote the use of historical and current interests, respectively (§4.2.2). PKA refers to pseudo-historical knowledge assignment (§4.3). First, CSN contributes to the stable accumulation of knowledge, as evidenced by the overall performance gains when it is included. Second, both CIE-H and CIE-C effectively enrich historical and current interests, respectively. This is supported by a decrease in RA and LA when each component is excluded. Third, PKA supports new-user adaptation by leveraging existing historical knowledge. Without it, both RA and LA drop, with a larger decline in RA. This suggests that failing to properly accommodate new users can negatively affect overall performance. Performance on new users is analyzed in the next section. Finally, the best H-mean is achieved when all modules are used together, highlighting the synergy among the proposed components.

5.3.3 Hyperparameter study. Table 6 presents the impact of top- K users in PKA on recommendation performance for new users. Here, $K=0$ indicates the case that PKA is not used. For all values of K , PKA shows better performance compared to not using it. This result indicates that PKA effectively addresses the lack of information in newly incoming users, thereby facilitating their adaptation.

Figure 4 provides results with varying the number (N) and length (L) of interest pools, where N improves diversity and L adds depth.

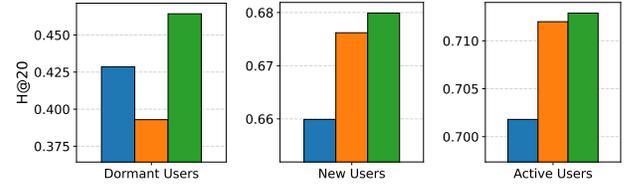


Figure 3: Hit@20 results on Gowalla across three user groups. (Blue: IMSR, Orange: SAIL-PIW, Green: CSTRec)

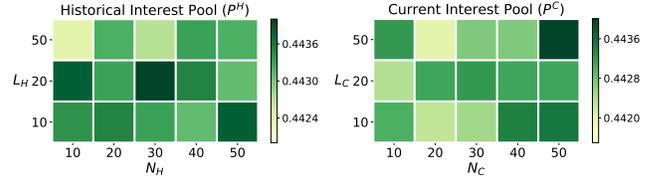


Figure 4: Impact of the number and length of interests.

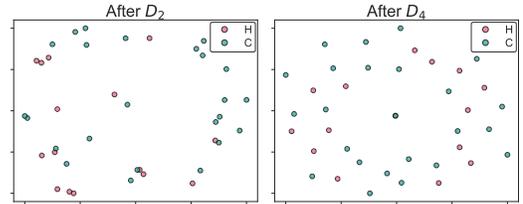


Figure 5: t-SNE visualization of interest pools.

We report the averaged H-mean from D_2 to D_4 . We find that allocating more capacity to the current interest pool yields the best results, highlighting the importance of capturing complex and diverse trends in incoming data blocks. In CSTRec, historical knowledge is leveraged to facilitate learning new interests, so fewer parameters suffice. Also, long-term user preferences encoded in historical pool tend to be less dynamic compared to the variety of transient interests in the current data block, which may also explain this tendency. Similar trends are also observed across other datasets.

5.3.4 Interest pool analysis. Figure 5 presents the t-SNE visualization results of historical and current interest pools for data blocks D_2 and D_4 . We conduct a visualization of interests averaged with respect to their lengths (i.e., L_H and L_C). We observe that as data progresses across blocks, each interest becomes more evenly distributed, indicating a progressive capture of more distinct knowledge. By capturing this diverse knowledge, CIE can provide tailored guidance for each user, compensating for forgotten knowledge and supplementing insufficient information in incoming sequences.

6 Conclusion

We introduce CSTRec, a transformer-based SR model capable of handling non-stationary data streams, which has been underexplored in previous literature. Building on the advantages of linear attention, we propose CSA with two novel components, CSN and CIE, enabling CSTRec to both retain historical knowledge and acquire current one over time. We also propose a pseudo-historical knowledge assignment strategy for new users to facilitate their adaptation. Our experiments show that CSTRec effectively captures the trajectory of user interests. We expect that CSTRec broadens the applicability of SR models to continuously changing environments.

Ethical Considerations

We believe this work does not pose significant negative societal impacts. Specifically, it does not raise any additional fairness, privacy, security, or safety concerns beyond standard practice.

Acknowledgement

This work was supported by ICT Creative Consilience Program through the IITP grant funded by the MSIT (IITP-2025-RS-2020-II201819) and Basic Science Research Program through the NRF funded by the Ministry of Education (NRF-2021R1A6A1A03045425), and 2025 High-Performance Computing Support Program.

References

- [1] Kian Ahrabian, Yishi Xu, Yingxue Zhang, Jiapeng Wu, Yuening Wang, and Mark Coates. 2021. Structure aware experience replay for incremental learning in graph-based recommender systems. In *CIKM*. 2832–2836.
- [2] Guohao Cai, Jieming Zhu, Quanyu Dai, Zhenhua Dong, Xiuqiang He, Ruiming Tang, and Rui Zhang. 2022. ReLoop: A Self-Correction Continual Learning Loop for Recommender Systems. In *SIGIR*. 2692–2697.
- [3] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 714–722.
- [4] Junsu Cho, Dongmin Hyun, SeongKu Kang, and Hwanjo Yu. 2021. Learning heterogeneous temporal patterns of user preference for timely recommendation. In *Proceedings of the Web Conference 2021*. 1274–1283.
- [5] Krzysztof Marcin Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking Attention with Performers. In *International Conference on Learning Representations*.
- [6] Djork-Arné Clevert. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).
- [7] Felipe Soares da Costa and Peter Dolog. 2019. Collective embedding for neural context-aware recommender systems. In *Proceedings of the 13th ACM conference on recommender systems*. 201–209.
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.
- [9] Jaime Hieu Do and Hady W Lauw. 2023. Continual Collaborative Filtering Through Gradient Alignment. In *RecSys*. 1133–1138.
- [10] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S Yu. 2022. Sequential recommendation via stochastic self-attention. In *Proceedings of the ACM web conference 2022*. 2036–2047.
- [11] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* 39, 1 (2020), 1–42.
- [12] Bowei He, Xu He, Yingxue Zhang, Ruiming Tang, and Chen Ma. 2023. Dynamically Expandable Graph Convolution for Streaming Recommendation. In *WWW*.
- [13] B Hidasi. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06939* (2015).
- [14] Martin Jakomin, Zoran Bosnić, and Tomaz Curk. 2020. Simultaneous incremental matrix factorization for streaming recommender systems. *Expert systems with applications* 160 (2020), 113685.
- [15] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.
- [16] Dahuin Jung, Dongjin Lee, Sunwon Hong, Hyemi Jang, Ho Bae, and Sungroh Yoon. 2023. New Insights for the Stability-Plasticity Dilemma in Online Continual Learning. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*.
- [17] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
- [18] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*. PMLR, 5156–5165.
- [19] Dongwan Kim and Bohyung Han. 2023. On the stability-plasticity dilemma of class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20196–20204.
- [20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.
- [21] Gyuseok Lee, SeongKu Kang, Wonbin Kweon, and Hwanjo Yu. 2024. Continual Collaborative Distillation for Recommender System. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1495–1505.
- [22] Soochan Lee, Jaehyeon Son, and Gunhee Kim. 2023. Recasting continual learning as sequence modeling. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*. 70433–70452.
- [23] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM conference on Information and Knowledge Management*. 1419–1428.
- [24] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.
- [25] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.
- [26] Langming Liu, Liu Cai, Chi Zhang, Xiangyu Zhao, Jingtong Gao, Wanyu Wang, Yifu Lv, Wenqi Fan, Yiqi Wang, Ming He, et al. 2023. Linrec: Linear attention mechanism for long-term sequential recommender systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 289–299.
- [27] Fei Mi, Xiaoyu Lin, and Boi Faltings. 2020. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *RecSys*. 408–413.
- [28] Tsenduren Munkhdalai, Manaal Faruqi, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143* (2024).
- [29] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 524–540.
- [30] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.
- [31] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong sequential modeling with personalized memorization for user response prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 565–574.
- [32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.
- [33] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of machine Learning research* 6, 9 (2005).
- [34] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. 2021. Deep learning for recommender systems: A Netflix case study. *AI Magazine* 42, 3 (2021), 7–18.
- [35] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.
- [36] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 17–22.
- [37] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems* 30 (2017).
- [38] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).
- [39] Ellen M Voorhees et al. 1999. The trec-8 question answering track report.. In *Trec*, Vol. 99. 77–82.
- [40] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).
- [41] Yuening Wang, Yingxue Zhang, and Mark Coates. 2021. Graph structure aware contrastive knowledge distillation for incremental learning in recommender systems. In *CIKM*. 3518–3522.
- [42] Yuening Wang, Yingxue Zhang, Antonios Valkanias, Ruiming Tang, Chen Ma, Jianye Hao, and Mark Coates. 2023. Structure aware incremental learning with personalized imitation weights for recommender systems. In *AAAI*. 4711–4719.
- [43] Zhikai Wang and Yanyan Shen. 2023. Incremental learning for multi-interest sequential recommendation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1071–1083.
- [44] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. 2022. Dualprompt:

- Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*. Springer, 631–648.
- [45] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 139–149.
- [46] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.
- [47] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM conference on recommender systems*. 328–337.
- [48] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James L Sharpnack. 2019. Stochastic shared embeddings: Data-driven regularization of embedding layers. *Advances in Neural Information Processing Systems* 32 (2019).
- [49] Yongji Wu, Lu Yin, Defu Lian, Mingyang Yin, Neil Zhenqiang Gong, Jingren Zhou, and Hongxia Yang. 2021. Rethinking Lifelong Sequential Recommendation with Incremental Multi-Interest Attention. arXiv:2105.14060 [cs.LG] <https://arxiv.org/abs/2105.14060>
- [50] Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. 2020. Graphsail: Graph structure aware incremental learning for recommender systems. In *CIKM*. 2861–2868.
- [51] Hyunsik Yoo, SeongKu Kang, Ruizhong Qiu, Charlie Xu, Fei Wang, and Hanghang Tong. 2025. Embracing plasticity: Balancing stability and plasticity in continual recommender systems. In *Proceedings of the 48th International ACM SIGIR conference on research and development in Information Retrieval*. 2092–2101.
- [52] Hyunsik Yoo, SeongKu Kang, and Hanghang Tong. 2025. Continual Recommender Systems. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 6857–6860.
- [53] Hyunsik Yoo, Ting-Wei Li, SeongKu Kang, Zhining Liu, Charlie Xu, Qilin Qi, and Hanghang Tong. 2025. Continual Low-Rank Adapters for LLM-based Generative Recommender Systems. *arXiv preprint arXiv:2510.25093* (2025).
- [54] Hyunsik Yoo, Zhichen Zeng, Jian Kang, Ruizhong Qiu, David Zhou, Zhining Liu, Fei Wang, Charlie Xu, Eunice Chan, and Hanghang Tong. 2024. Ensuring user-side fairness in dynamic recommender systems. In *Proceedings of the ACM Web Conference 2024*. 3667–3678.
- [55] Tong Yu, Ole J Mengshoel, Alvin Jude, Eugen Feller, Julien Forgeat, and Nimish Radia. 2016. Incremental learning for matrix factorization in recommender systems. In *2016 IEEE International conference on big data (Big Data)*. IEEE, 1056–1063.
- [56] Zhenrui Yue, Yueqi Wang, Zhankui He, Huimin Zeng, Julian McAuley, and Dong Wang. 2024. Linear recurrent units for sequential recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 930–938.
- [57] Peiyan Zhang and Sunghun Kim. 2023. A survey on incremental update for neural recommender systems. *arXiv preprint arXiv:2303.02851* (2023).
- [58] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 1893–1902.
- [59] Jieming Zhu, Guohao Cai, Junjie Huang, Zhenhua Dong, Ruiming Tang, and Weinan Zhang. 2023. ReLoop2: Building Self-Adaptive Recommendation Models via Responsive Error Compensation Loop. In *KDD*.