VAMIRec: Value-Aware Memory Intervention for Continual Recommendation

Xudong Zhao[1], **Xiaolong Xu**[1,*], **Haolong Xiang**[1,2], **Jiaqiang Zhang**[1,2], **Xuyun Zhang**[3], **Lianyong Qi**[4], **Wanchun Dou**[2], **Amin Beheshti**[3]
[1]School of Software, Nanjing University of Information Science and Technology, P.R. China
[2]State Key Lab. of Novel Software Technology, Nanjing University, P.R. China
[3]School of computing, Macquarie University, Australia
[4]College of Computer Science and Technology, China University of Petroleum (East China), P.R. China

Abstract

Continual recommendation captures dynamic user interests amid sequential interactions, enabling models to adapt to new observations while preserving prior valuable knowledge. Existing continual recommendation methods primarily preserve or reuse historical knowledge through replay, distillation, or regularization to mitigate forgetting during incremental updates. However, noisy or obsolete inherited representations render indiscriminate preservation detrimental by inducing memory conflicts with current stage preferences. Moreover, rapid shifts in user interests require timely and selective memory updates, while coarse global mechanisms fail to provide such fine-grained responsiveness. To address these issues, we propose VAMIRec, a value-aware memory intervention framework for continual recommendation. VAMIRec first constructs a memory state by separating inherited user memory into a slow anchor component and a fast-changing drift component. We further design a value-aware action evaluation module to generate three candidate operations, and estimate conservative action values by accounting for uncertainty and intervention cost. Finally, we develop a policy-guided memory intervention module that distills the training-time action supervision into an inference-safe policy and applies the selected memory action before all-item ranking. Extensive experiments on three real-world datasets demonstrate that VAMIRec consistently outperforms the state-of-the-art baselines.

1 Introduction

Continual recommender systems operate in dynamic environments where user interactions arrive sequentially and user interests evolve over time. Recent advances in recommendation have substantially improved ranking effectiveness by learning expressive user and item representations from implicit feedback, collaborative signals, and user-item graphs [1–3]. Sequential recommenders further model ordered user behaviors to capture short-term preference transitions from interaction sequences [4–6]. However, continual recommendation imposes an additional requirement: the model must repeatedly adapt to newly arriving interactions while deciding how to reuse user representations inherited from previous stages [7, 8]. This setting raises a central challenge: how to reconcile current-stage interactions with historical user memory accumulated from previous stages [9, 10]. Current-stage interactions are sensitive to short-term intent, whereas historical memory often captures stable long-term preference. Crucially, historical memory is not uniformly reliable [11, 12]. As user interests and contexts evolve, inherited representations may remain useful, accumulate noise or obsolete signals, or even interfere with current prediction [13–15]. This observation reframes

*Corresponding author. Email: xlxu@ieee.org

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

memory reuse as a value-aware decision problem: inherited user memory should be evaluated against current-stage evidence and selectively retained, revised, or attenuated according to its contribution to recommendation [16, 17].

Most existing continual recommendation methods address this problem from the perspective of forgetting prevention. Replay-based methods store or reuse historical interactions to reduce catastrophic forgetting during incremental learning [18, 19]. Other approaches learn continual user representations by reusing user-specific parameters or modeling task relations across stages [20, 21]. Graph-based methods preserve historical graph structures or distill graph-aware knowledge from previous stages [22, 23], and recent approaches further introduce personalized imitation, collaborative distillation, or stability–plasticity control during continual updates [24, 9, 25]. Although replay and representation-based continual recommendation methods improve stability under incremental updates, these methods often rely on coarse global mechanisms for preserving, reusing, or updating historical knowledge [25, 26]. Such coarse treatment becomes inadequate when user interests shift rapidly while only part of the inherited representation becomes obsolete. Some components of an inherited representation may conflict with current-stage interests, while others may still encode useful long-term preference [27]. As a result, continual recommendation requires a more selective mechanism that can decide whether historical memory should be preserved, locally revised, or gradually attenuated at each stage.

Recent progress on model editing shows that internal knowledge can be modified without full model retraining [28, 29]. Subsequent studies further locate, edit, or fine-tune internal representations in a targeted manner [30–33]. Yet continual recommendation introduces a distinct setting. The object to be managed is not a generic hidden state or a parameter block, but a user-specific historical representation that will be repeatedly reused for future ranking [34]. This setting imposes two requirements. First, memory management should explicitly assess the stage-dependent value of inherited user memory rather than assuming that old knowledge is always beneficial. Second, intervention should remain inference-safe, which means that the decision rule should depend only on user-side states at test time rather than on target items or sampled negatives available only during training. These observations motivate a value-aware and constrained formulation of historical memory intervention.

Motivated by this perspective, we propose **VAMIRec**, a value-aware memory intervention framework for continual recommendation. VAMIRec formulates historical representation management as a constrained decision problem over three memory actions: retain, edit, and decay. It first constructs a memory state that separates inherited user memory into a stable anchor component and a transient drift component, and then estimates conservative action values from training-time ranking feedback while accounting for uncertainty and intervention cost. The resulting action supervision is distilled into an inference-safe policy that selects a memory action before all-item ranking, where sparse budget-constrained editing enables local correction of stale or conflicting memory while preserving stable long-term preferences.

Our contributions are summarized as follows.

- We propose **VAMIRec**, a value-aware memory intervention framework for continual recommendation under evolving user interests.
- We formulate historical memory reuse as an inference-safe action-value learning problem over retain, edit, and decay decisions.
- We provide theoretical analyses for slow-anchor stability, bounded sparse editing, and quantile-based edit-rate control.
- Experiments on three real-world datasets demonstrate the effectiveness and generality of VAMIRec. The implementation code is available `https://github.com/bolunxier123/VAMIRec`.

2 Problem Formulation

We consider a continual recommendation setting where user–item interactions arrive in T consecutive stages, $\mathcal{D}^{(0)}, \mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(T-1)}$. At stage t, the recommender receives the current-stage data $\mathcal{D}^{(t)}$ and updates its parameters for the current recommendation task. For user u, let $q_u^{(t)} \in \mathbb{R}^d$ denote the current user embedding produced by a backbone encoder from the current-stage evidence, and let $e_i \in \mathbb{R}^d$ denote the item embedding of item i. The model may also inherit user-side historical

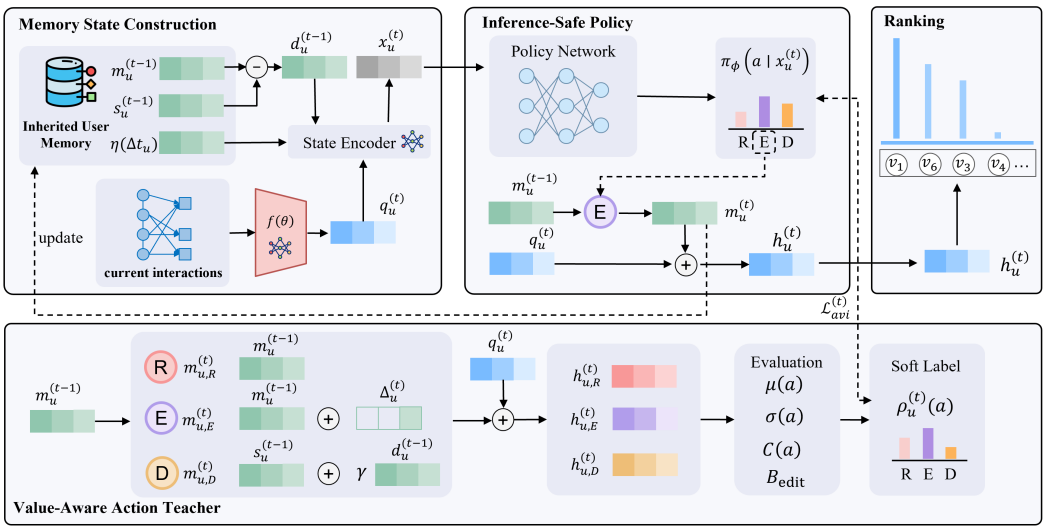


Figure 1: Overview of VAMIRec. VAMIRec first constructs a compact memory state from current evidence, stable preference, drift, and memory age. A value-aware action teacher compares retain, edit, and decay candidates using ranking feedback with uncertainty, cost, and edit-budget control. The learned inference-safe policy selects a memory action from user-side states and applies the intervention before all-item ranking.

information from previous stages, which is used together with current evidence to form the final user embedding.

The goal of continual recommendation is to learn a stage-wise recommender that adapts to newly observed interactions while still making effective use of historical user information. This problem is challenging because user interests may evolve over time, and historical representations learned in previous stages may become partially stale under changing contexts and non-stationary data distributions. Therefore, the model should improve current-stage recommendation performance without blindly relying on historical information that may no longer match the present user preference.

For recommendation at stage t, the final user embedding is denoted by $h_u^{(t)}$. The prediction score of item i for user u is computed as

$$\hat{y}_{u,t,i} = h_u^{(t)\top} e_i. \tag{1}$$

During training, for each user instance, we follow the standard sampled ranking protocol with one observed positive item i^+ and M sampled negative items $\{i_j^-\}_{j=1}^M$. During evaluation, we adopt all-item ranking, where all items that the user has not interacted with are treated as candidates.

3 Method

This section presents **VAMIRec**, a value-aware memory intervention framework for continual recommendation. The key idea is to treat inherited user memory as a stage-dependent state whose utility should be reassessed before reuse, rather than uniformly preserved. VAMIRec formulates historical memory reuse as a selective intervention problem over three actions: retain, edit, and decay, which respectively preserve reliable memory, correct conflict-prone memory, and attenuate stale drift.

Since the appropriate action is not directly observed, VAMIRec uses training-time ranking feedback to construct a value-aware action teacher and distills it into an inference-safe policy that depends only on user-side states. As illustrated in Figure 1, VAMIRec consists of memory state construction, value-aware action teacher, inference-safe policy, and memory update.

3.1 Memory State Construction

Historical memory must be represented in a form that exposes both what should be preserved and what may need intervention. A single inherited memory embedding is not sufficient for this

purpose, because it mixes relatively stable preference with stage-specific drift. When current evidence disagrees with historical memory, the model should know whether the discrepancy reflects obsolete transient behavior or useful long-term preference.

To separate these two effects, VAMIRec maintains a working memory embedding $m_u^{(t-1)}$ and a slow anchor embedding $s_u^{(t-1)}$ for each user. The working memory stores the inherited representation used for current-stage prediction, while the slow anchor evolves more conservatively and provides a stable preference reference. We define the drift embedding as

$$d_u^{(t-1)} = m_u^{(t-1)} - s_u^{(t-1)}. \tag{2}$$

The drift embedding captures the fast-changing residual of historical memory around the stable anchor. Appendix A.1 justifies the slow anchor through a stationary-variance analysis under a noisy-observation model.

The intervention policy should receive a compact state rather than redundant memory features. Since $m_u^{(t-1)} = s_u^{(t-1)} + d_u^{(t-1)}$, directly concatenating the working memory, anchor, and drift repeatedly exposes the same information. VAMIRec therefore constructs the user-side state as

$$x_u^{(t)} = \mathrm{StateEnc}\left(q_u^{(t)}, s_u^{(t-1)}, d_u^{(t-1)}, \eta(\Delta t_u)\right), \tag{3}$$

where $q_u^{(t)}$ is the current user embedding, $\Delta t_u = t - t_u^{\mathrm{last}}$ measures the memory age, and $\eta(\cdot)$ is a normalized age feature. The anchor and drift summarize inherited memory, while the age feature captures temporal inactivity that cannot be inferred from embeddings alone.

Conflict should guide how much memory can be edited, but it need not be an additional policy input. We therefore keep the policy state compact and use conflict only to bound the sparse edit radius. Specifically, we compute

$$c_u^{(t)} = \left\| \mathrm{Norm}(q_u^{(t)}) - \mathrm{Norm}(s_u^{(t-1)} + d_u^{(t-1)}) \right\|_2,$$

which measures the mismatch between current evidence and reconstructed inherited memory. The age normalization, memory initialization, and implementation of $\mathrm{StateEnc}(\cdot)$ are provided in Appendix A.1.

The memory state must also avoid target leakage during training. If the state contains the target interaction or future interactions within the same stage, the learned intervention policy can become artificially strong. Thus, $q_u^{(t)}$ is computed without using the target interaction being predicted. In our graph-based instantiation, the supervised edge is masked when constructing the training state. During optimization at stage t, inherited memory embeddings $m_u^{(t-1)}$ and $s_u^{(t-1)}$ are frozen and written back only after the stage finishes. Appendix A.1 provides the leakage-avoidance argument.

3.2 Value-Aware Action Teacher

Once the memory state is constructed, the model needs a small and interpretable action space for deciding how to reuse inherited memory. VAMIRec considers three actions that correspond to typical states of historical memory: reliable memory should be retained, locally conflicting memory should be edited, and stale drift should be decayed. The action-specific memory candidates are defined as

$$\tilde{m}_{u,a}^{(t)} = \begin{cases} m_u^{(t-1)}, & a = \mathrm{retain}, \\ m_u^{(t-1)} + \Delta_u^{(t)}, & a = \mathrm{edit}, \\ s_u^{(t-1)} + \gamma_u^{(t)} d_u^{(t-1)}, & a = \mathrm{decay}. \end{cases} \tag{4}$$

Retain keeps inherited memory unchanged. Edit adds a sparse correction $\Delta_u^{(t)}$ to conflict-prone dimensions. Decay suppresses the drift embedding through $\gamma_u^{(t)}$ while preserving the anchor. The sparse low-rank editor for $\Delta_u^{(t)}$ and the decay coefficient are given in Appendix A.3.

To compare memory actions fairly, each candidate should be evaluated under the same current evidence. For action a, VAMIRec forms the action-specific user embedding as

$$h_{u,a}^{(t)} = \mathrm{Norm}\left(q_u^{(t)} + \tilde{m}_{u,a}^{(t)}\right). \tag{5}$$

Thus, the difference among candidates comes from the intervention applied to inherited memory, rather than from changes in the current encoder output.

Since no ground-truth action label is available, VAMIRec constructs training-time supervision from the ranking objective itself. For action a, positive item i^+, and sampled negatives $\{i_j^-\}_{j=1}^M$, we define the pairwise ranking utility as

$$\ell_{u,j}^{(t)}(a) = \log \sigma \left(h_{u,a}^{(t)\top} e_{i^+} - h_{u,a}^{(t)\top} e_{i_j^-} \right). \tag{6}$$

This utility measures how the current model's ranking objective changes when a candidate memory action is applied to the same user-side state. It is used only to build action supervision during training. Appendix A.2 provides the detailed interpretation.

A teacher that simply chooses the action with the largest empirical utility can be unstable. The utility is estimated from finitely sampled negatives, and aggressive interventions may obtain short-term gains by making unnecessarily large changes to reusable memory. VAMIRec therefore defines a conservative action value:

$$Q_u^{(t)}(a) = \hat{\mu}_u^{(t)}(a) - \kappa_r \frac{\hat{\sigma}_u^{(t)}(a)}{\sqrt{M}} - \lambda_c C_u^{(t)}(a), \quad V_u^{(t)}(a) = Q_u^{(t)}(a) - Q_u^{(t)}(\text{retain}), \tag{7}$$

where $\hat{\mu}_u^{(t)}(a)$ and $\hat{\sigma}_u^{(t)}(a)$ are the mean and standard deviation of $\ell_{u,j}^{(t)}(a)$ over sampled negatives. The uncertainty term penalizes unstable action utility, while the intervention cost $c_u^{(t)}(a)$ regularizes nontrivial memory modifications, assigning zero cost to retain and larger costs to edit or decay according to their intervention strength. The retain-normalized value $V_u^{(t)}(a)$ therefore measures the marginal benefit of an intervention relative to preserving the inherited memory unchanged.

Edit requires additional control because it is more flexible than retain and decay. Although editing can correct local conflicts, unconstrained editing may corrupt historical memory that will be reused in later stages. VAMIRec therefore applies an edit-budget projection before teacher normalization. Only users with the largest edit advantages in a mini-batch are allowed to receive finite edit value. Let $\bar{V}_u^{(t)}(a)$ denote the projected value. The teacher distribution is computed as

$$\rho_u^{(t)}(a) = \text{softmax} \left(\frac{\text{sg}(\bar{V}_u^{(t)}(a))}{\tau} \right), \tag{8}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. The resulting distribution serves as a soft action label for the inference policy. The budget projection rule and its constrained assignment interpretation are given in Appendix A.2.

3.3 Inference-Safe Policy

The action teacher cannot be directly used at inference because it depends on the positive item and sampled negatives. In all-item ranking, the recommender must select a memory action before scoring candidate items. VAMIRec therefore distills the teacher distribution into an inference-safe policy:

$$\pi_\phi^{(t)}(a) = \pi_\phi(a \mid x_u^{(t)}) = \text{softmax} \left(g_\phi(x_u^{(t)}) \right). \tag{9}$$

The policy only observes the user-side state and does not use target items, sampled negatives, or evaluation batch statistics. Once an action is selected, the resulting user embedding is shared by all candidate items.

The edit budget also needs an inference-safe deployment rule. The training-time budget is imposed through mini-batch projection, but applying the same projection at test time would make the decision depend on the evaluation batch. To avoid this dependence, VAMIRec calibrates an edit threshold using validation user states or a running stage-wise quantile of edit-logit advantages. This threshold controls the expected edit rate while keeping the deployed decision independent of candidate items and evaluation batches. The deployment rule and expected edit-rate analysis are given in Appendix A.4.

When the policy selects edit, the memory change should remain local and bounded. VAMIRec therefore applies the sparse low-rank editor described in Appendix A.3, which combines a low-rank bottleneck, a straight-through top-k mask, and conflict-dependent clipping.

3.4 Learning Objective and Memory Update

The final representation should correspond to a concrete intervention rather than a soft average of all memory actions. VAMIRec therefore uses a hard action in the forward pass and preserves gradients through the soft policy with a straight-through estimator. The final user embedding is

$$h_u^{(t)} = \mathrm{Norm}\left(q_u^{(t)} + \sum_{a\in\mathcal{A}} \tilde{\pi}_u^{(t)}(a)\tilde{m}_{u,a}^{(t)}\right), \tag{10}$$

where $\tilde{\pi}_u^{(t)}$ denotes the straight-through action indicator. Since the action is selected before item scoring, the same $h_u^{(t)}$ is used for all candidate items. The detailed estimator is given in Appendix A.5.

The learning objective should jointly optimize recommendation accuracy and deployable action selection. VAMIRec is trained with a compact two-term objective:

$$\mathcal{L}^{(t)} = \mathcal{L}_{\mathrm{rec}}^{(t)} + \lambda_{\mathrm{avi}} \frac{1}{|\mathcal{B}|} \sum_{u\in\mathcal{B}} \mathrm{KL}\left(\mathrm{sg}(\rho_u^{(t)}) \| \pi_\phi^{(t)}\right). \tag{11}$$

The recommendation loss trains the intervened user embedding for ranking, while the imitation term transfers the teacher's action preference to the inference-safe policy. Sparse editing, conservative values, and edit-budget control are handled inside the intervention and teacher construction, rather than added as separate loss terms. The BPR instantiation of $\mathcal{L}_{\mathrm{rec}}^{(t)}$ is given in Appendix A.5.

The memory written to the next stage should incorporate useful current-stage intervention without contaminating the stable anchor. After stage t is optimized, VAMIRec updates the working memory and anchor as

$$m_u^{(t)} = \beta_m m_u^{(t-1)} + (1-\beta_m)\bar{m}_u^{(t)}, \quad s_u^{(t)} = \beta_s s_u^{(t-1)} + (1-\beta_s)\mathrm{sg}\left(\mathrm{Norm}(q_u^{(t)})\right). \tag{12}$$

The working memory absorbs the intervened memory used at the current stage, while the anchor is updated from current evidence rather than from the intervened memory. This prevents edit or decay decisions from directly contaminating the slow preference reference. We use $\beta_s > \beta_m$, so the anchor evolves more slowly than the working memory. At stage 0, no historical memory is available and VAMIRec reduces to the backbone recommender. The definition of $\bar{m}_u^{(t)}$ and the complexity analysis are provided in Appendix A.5.

4 Experiment

4.1 Experimental Settings

Datasets. We evaluate VAMIRec on three real-world datasets from different recommendation scenarios: Amazon (CDs and Vinyls), Amazon (Video Games), and Gowalla. For preprocessing, we follow common implicit-feedback settings and construct continual stages according to interaction timestamps. More details about the datasets and preprocessing are provided in Appendix B.1.

Baselines. We compare VAMIRec with seven baselines: three general continual adaptation strategies, including **Fine-tune**, **Full-Retrain**, and **Replay** [18], and four continual recommendation methods, including **GraphSAIL** [22], **LWC-KD** [23], **LWC-KD-PIW** [24], **CCD** [9], and **PISA** [25]. Detailed descriptions are provided in Appendix B.2.

Implementation Details. To assess the effectiveness of each method, we initially identified the best hyperparameters with a single seed applied to the validation set. Subsequently, each method was trained with the selected hyperparameters across five different random seeds. More details about the datasets and preprocessing can be seen in Appendix B.3.

Evaluation Metrics. We adopt the all-item ranking protocol, where all items that a user has not interacted with are treated as candidate items. Top-N recommendation accuracy is evaluated by Recall@N and NDCG@N, with $N \in \{10, 20\}$. All results are averaged over the T incremental data blocks [35, 36].

Additional backbone generalization results, including both graph-based and sequential recommendation backbones, are reported in Appendix C.1.

Table 1: Overall performance comparison on three continual recommendation benchmarks. The best results are highlighted in bold and the second-best results are underlined. "*" indicates statistically significant improvements over the best baseline under a two-sided t-test with $p < 0.05$. All values are reported as percentages.

Model	Amazon (CDs and Vinyls)				Amazon (Video Games)				Gowalla			
	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20
Fine-tune	0.62	0.38	1.34	0.58	0.81	0.50	1.37	0.67	2.07	1.48	3.00	1.76
Full-Retrain	0.78	0.48	1.31	0.62	1.31	0.85	2.33	1.15	2.59	1.92	3.90	2.34
Replay	0.81	0.51	1.43	0.71	1.33	0.86	2.16	1.11	2.56	1.84	3.89	2.27
GraphSAIL	0.73	0.45	1.28	0.56	1.21	0.72	2.10	1.02	1.96	1.43	2.95	1.72
LWC-KD	0.83	0.54	1.40	0.68	1.26	0.85	1.98	0.95	2.41	1.71	3.62	2.11
LWC-KD-PIW	0.86	0.51	1.47	0.69	1.36	0.86	2.25	1.13	2.80	2.08	4.12	2.46
CCD	0.95	0.58	1.55	0.80	1.49	0.99	2.27	1.21	2.84	2.10	4.18	2.50
PISA	0.89	0.62	1.58	0.83	1.55	1.03	2.20	1.22	2.90	2.16	4.21	2.47
VAMIRec	**1.06***	**0.66***	**1.81***	**0.88***	**1.76***	**1.11***	**2.68***	**1.39***	**3.06***	**2.21***	**4.48***	**2.65***

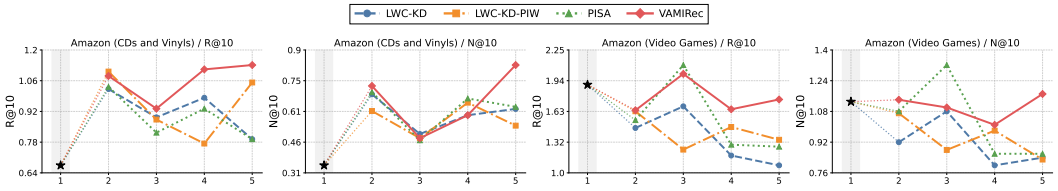


Figure 2: Stage-wise performance evolution on Amazon (CDs and Vinyls) and Amazon (Video Games). VAMIRec maintains more stable and stronger performance across later stages than continual recommendation baselines.

4.2 Overall Performance

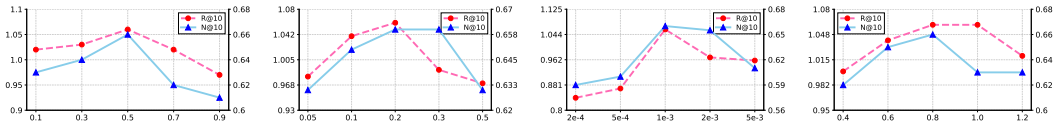
Table 1 reports the overall performance on three continual recommendation benchmarks. For a fair comparison, all methods are trained and evaluated with the same LightGCN backbone, continual data splits, negative sampling setting, and all-item ranking protocol. VAMIRec achieves the best results across all datasets and evaluation metrics, showing that the proposed memory intervention framework is effective under different data sparsity levels and domain characteristics.

The corrected Full-Retrain results show that cumulative retraining is a competitive reference, since using all observed interactions can provide stronger historical signals than simple fine-tuning or replay in several cases. However, Full-Retrain still falls behind VAMIRec, indicating that simply accumulating historical data is not sufficient when user interests evolve over time. This supports our motivation that historical representations should not be uniformly reused, but should be selectively assessed and intervened according to their current-stage utility.

Compared with specialized continual recommendation baselines such as LWC-KD, LWC-KD-PIW, CCD, and PISA, VAMIRec also obtains consistent improvements. These methods mainly preserve or distill historical knowledge to balance stability and plasticity, whereas VAMIRec explicitly decomposes inherited memory into stable preference and transient drift, and then learns whether to retain, edit, or decay the memory under an inference-safe policy. The consistent gains suggest that value-aware memory evaluation and constrained intervention provide a finer-grained mechanism for adapting to evolving user preferences.

4.3 Stage-wise Performance Evolution

Figure 2 presents the stage-wise performance evolution of different methods on Amazon (CDs and Vinyls) and Amazon (Video Games). We observe that all methods start from the same performance at the first stage, since no historical memory is available at initialization and the models reduce to the same backbone recommender. As training proceeds, clear differences emerge across methods. On both datasets, VAMIRec exhibits a more favorable performance trajectory than the competing baselines, especially in the later stages. Compared with LWC-KD, LWC-KD-PIW, and PISA, VAMIRec not only achieves stronger stage-wise peaks, but also maintains a more stable trend without suffering from severe performance degradation after adaptation. This pattern is particularly



(a) λ_{avi} (b) B_{edit} (c) lr (d) τ

Figure 3: Hyperparameter sensitivity analysis of VAMIRec on the Amazon (CDs and Vinyls) dataset. We evaluate the effects of the imitation weight λ_{avi}, edit budget B_{edit}, learning rate lr, and teacher temperature τ.

Table 2: Ablation study of VAMIRec on three datasets. The boldface refers to the highest score and the underline indicates the next best result of the models.

	Amazon (CDs and Vinyls)				Amazon (Video Games)				Gowalla			
Model	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20
w/o MD	1.04	0.65	1.75	0.84	1.60	1.05	2.51	1.32	2.93	2.04	4.36	2.43
w/o Decay	0.95	0.62	1.62	0.82	1.45	0.98	2.42	1.27	2.60	1.90	3.78	2.27
w/o Value	1.02	0.63	1.73	0.84	1.60	0.96	2.58	1.26	3.00	2.08	4.29	2.30
w/o CC	0.93	0.59	1.51	0.77	1.47	0.95	2.38	1.22	2.54	1.96	3.70	2.16
w/o Policy	0.99	0.59	1.72	0.82	1.49	0.95	2.49	1.22	2.84	2.01	3.96	2.47
VAMIRec	**1.06**	**0.66**	**1.81**	**0.88**	**1.76**	**1.11**	**2.68**	**1.39**	**3.06**	**2.21**	**4.48**	**2.65**

evident on Amazon (CDs and Vinyls), where VAMIRec continues to improve in the later stages while PISA shows noticeable fluctuations, suggesting that our method is better at preserving useful historical information while suppressing stale memory. A similar phenomenon can be observed on Amazon (Video Games), where VAMIRec consistently retains stronger performance in the final stages, indicating better adaptation under continual distribution shifts. Overall, these results show that VAMIRec improves not only the final average performance, but also the entire adaptation trajectory, leading to a more stable and effective continual recommendation process.

4.4 Hyperparameter Sensitivity Analysis

Figure 3 presents the hyperparameter sensitivity of VAMIRec on the Amazon (CDs and Vinyls) dataset. As shown in Figure 3(a), the imitation weight λ_{avi} achieves the best performance at a moderate value. A small λ_{avi} provides insufficient supervision for transferring the teacher's action preference to the inference policy, whereas an overly large value may overemphasize action imitation and weaken the ranking objective. Figure 3(b) shows a similar trend for the edit budget B_{edit}. A limited budget makes the model overly conservative and prevents useful memory correction, while an excessive budget may edit too many user memories and disturb stable long-term preferences. For the learning rate in Figure 3(c), a small value slows down adaptation, whereas a large value leads to unstable joint optimization. Finally, Figure 3(d) shows that the teacher temperature τ also requires a balanced setting: an overly sharp distribution produces unstable action supervision, while an overly smooth distribution weakens the distinction among retain, edit, and decay. Overall, these results indicate that VAMIRec benefits from balanced intervention control, where the teacher provides informative but not overconfident supervision and memory editing is applied selectively under a bounded budget.

4.5 Ablation Study

The ablation results are reported in Table 2. We evaluate five variants along the pipeline of VAMIRec: *w/o MD* removes the anchor–drift memory decomposition, *w/o Decay* removes the decay action for stale historical representations, *w/o Value* removes the action-value evaluation module, *w/o CC* removes the conservative control terms in the action teacher, and *w/o Policy* removes the inference-safe intervention policy. The full model consistently achieves the best performance across all datasets and metrics, suggesting that these components contribute complementary benefits.

Removing memory decomposition leads to consistent degradation, indicating that a single inherited memory embedding is insufficient to separate stable preference from transient drift. This supports the need for a structured memory state before intervention. The performance drop of *w/o Decay* further

shows that stale historical signals should not be uniformly retained; by attenuating obsolete drift while preserving the stable anchor, the decay branch provides a controlled mechanism for handling outdated memory. The variants *w/o Value* and *w/o CC* highlight the importance of value-aware and conservative action evaluation. Without explicit comparison among retain, edit, and decay candidates, the model loses the ability to select interventions according to their estimated ranking utility; without uncertainty penalization, intervention cost, and edit-budget projection, the teacher may favor overly aggressive interventions that improve short-term utility but damage reusable historical memory. Finally, the degradation of *w/o Policy* shows that training-time action values alone are not sufficient, and an inference-safe policy is necessary to convert action supervision into deployable user-side decisions before all-item ranking. Overall, these results validate the main design of VAMIRec: effective continual recommendation requires structured memory construction, value-aware action comparison, conservative intervention control, and inference-safe memory reuse.

5 Related Work

Traditional and Sequential Recommendation. Recommendation methods have evolved from static collaborative filtering to dynamic preference modeling. BPR provides a classical pairwise ranking objective for implicit-feedback top-N recommendation, establishing a foundation for learning from historical interactions [37]. Building on this paradigm, neural collaborative filtering improves user–item matching with nonlinear interaction functions [1]. While these methods mainly treat interactions as unordered feedback, SASRec models behavior sequences with self-attention to capture short-term preference transitions [4]. In parallel, NGCF exploits the user–item interaction graph to propagate high-order collaborative signals [2]. LightGCN further simplifies graph propagation by removing unnecessary nonlinear transformations, making graph recommendation more effective and efficient [3]. More recently, Interest Changes explicitly models the life cycle of user interests, showing that historical preferences may emerge, stabilize, or decline over time [15]. IDURL further learns interest-drift-aware user representations for sequential recommendation under evolving preferences [12]. However, those works mostly assume static or single-stage training, leaving the reuse of inherited representations under current–historical conflicts underexplored.

Continual Recommendation. Continual recommendation extends recommendation to sequentially arriving interactions, where the model must adapt to new data while retaining useful historical knowledge. Experience replay provides a general continual learning strategy by reusing past examples to reduce catastrophic forgetting during incremental updates [18]. Following this idea in recommendation, GraphSAIL preserves self-information, local neighborhood structure, and global graph structure from previous interaction graphs [22]. Instead of preserving graph structures, Conure learns continual user representations through user-specific parameter reuse and pruning [20]. LWC-KD shifts the focus from parameter reuse to knowledge preservation by transferring historical graph knowledge through contrastive distillation [23]. TERACON further improves continual user representation learning by modeling task relations across stages [21]. LWC-KD-PIW refines graph-based distillation with personalized imitation weights, allowing different users to receive different strengths of historical knowledge transfer [24]. CCD advances this distillation line by co-evolving teacher and student recommenders during continual updates [9]. PISA explicitly studies the stability–plasticity trade-off for users with preference shifts, highlighting the need to balance historical preservation and current adaptation [25]. DITTO introduces dynamic time-aware continual user representation learning to handle item-distribution shifts over time [38]. The Federated Continuous Recommendation extends the Continuous Recommendation to Decentralized DataStreams to preserve privacy [39]. CSTRec studies continual sequential recommendation from behavior streams, improving sequential recommenders under continuously changing user interests [40]. Despite this progression, existing methods mainly preserve, transfer, or distill historical knowledge once it is inherited. In contrast, VAMIRec explicitly evaluates the stage-dependent value of inherited user memory and learns an inference-safe policy to selectively retain, edit, or decay memory before all-item ranking.

6 Conclusion

We presented VAMIRec, a value-aware memory intervention framework for continual recommendation. VAMIRec reformulates historical representation reuse as a selective intervention problem, where inherited user memory is reassessed and acted upon through retain, edit, and decay deci-

sions. By combining structured memory decomposition, conservative action-value evaluation, and an inference-safe intervention policy, VAMIRec enables controlled adaptation to evolving user interests while preserving useful long-term preference signals. Experiments on three real-world benchmarks demonstrate consistent improvements over strong continual recommendation baselines, with further analyses validating the effectiveness and generality of the proposed framework. These results highlight the importance of moving beyond uniform forgetting prevention toward value-aware memory management in dynamic recommendation.

Acknowledgments

This work was partially supported by the Fundamental Research Funds for the General Project for Basic Scientific Research of Higher Education Institutions in Jiangsu Province, under Grant No. 25KJB520028, the National Natural Science Foundation of China, under Grant No. 62372242, and the State Key Laboratory for Novel Software Technology, Nanjing University, under grant no. KFKT2026B46.

References

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[2] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[3] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[4] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE international conference on data mining (ICDM)*. IEEE, 2018, pp. 197–206.

[5] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 1441–1450.

[6] X. Xu, H. Dong, H. Xiang, X. Hu, X. Li, X. Xia, X. Zhang, L. Qi, and W. Dou, "C2lrec: Causal contrastive learning for user cold-start recommendation with social variables," *ACM Transactions on Information Systems*, vol. 43, no. 6, pp. 1–28, 2025.

[7] H. Yoo, S. Kang, and H. Tong, "Continual recommender systems," in *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, 2025, pp. 6857–6860.

[8] X. Xu, X. Zhao, H. Xiang, X. Zhang, W. Shen, H. Hu, and L. Qi, "Hpserec: A hierarchical partitioning and stepwise enhancement framework for long-tailed sequential recommendation," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[9] G. Lee, S. Kang, W. Kweon, and H. Yu, "Continual collaborative distillation for recommender system," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 1495–1505.

[10] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE transactions on pattern analysis and machine intelligence*, vol. 46, no. 8, pp. 5362–5383, 2024.

[11] R. Tiwari, K. Killamsetty, R. Iyer, and P. Shenoy, "Gcr: Gradient coreset based replay buffer selection for continual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 99–108.

[12] X. Lin, W. Pan, and Z. Ming, "Towards interest drift-driven user representation learning in sequential recommendation," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025, pp. 1541–1551.

[13] X. Yang, H. Yu, X. Gao, H. Wang, J. Zhang, and T. Li, "Federated continual learning via knowledge fusion: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 8, pp. 3832–3850, 2024.

[14] Z. Wang, Z. Zhang, C.-Y. Lee, H. Zhang, R. Sun, X. Ren, G. Su, V. Perot, J. Dy, and T. Pfister, "Learning to prompt for continual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 139–149.

[15] Y. Cai, J. Hou, Y. Zhu, and Y. Nie, "Interest changes: Considering user interest life cycle in recommendation system," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025, pp. 2592–2596.

[16] X. Xu, H. Dong, L. Qi, X. Zhang, H. Xiang, X. Xia, Y. Xu, and W. Dou, "Cmclrec: Cross-modal contrastive learning for user cold-start sequential recommendation," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 1589–1598.

[17] L. Wang, J. Zhang, H. Yang, Z.-Y. Chen, J. Tang, Z. Zhang, X. Chen, Y. Lin, H. Sun, R. Song *et al.*, "User behavior simulation with large language model-based agents," *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–37, 2025.

[18] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," *Advances in neural information processing systems*, vol. 32, 2019.

[19] P. Bruno, A. Quarta, and F. Calimeri, "Continual learning in medicine: A systematic literature review," *Neural Processing Letters*, vol. 57, no. 1, p. 2, 2025.

[20] F. Yuan, G. Zhang, A. Karatzoglou, J. Jose, B. Kong, and Y. Li, "One person, one model, one world: Learning continual user representation without forgetting," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 696–705.

[21] S. Kim, N. Lee, D. Kim, M. Yang, and C. Park, "Task relation-aware continual user representation learning," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 1107–1119.

[22] Y. Xu, Y. Zhang, W. Guo, H. Guo, R. Tang, and M. Coates, "Graphsail: Graph structure aware incremental learning for recommender systems," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2861–2868.

[23] Y. Wang, Y. Zhang, and M. Coates, "Graph structure aware contrastive knowledge distillation for incremental learning in recommender systems," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3518–3522.

[24] Y. Wang, Y. Zhang, A. Valkanas, R. Tang, C. Ma, J. Hao, and M. Coates, "Structure aware incremental learning with personalized imitation weights for recommender systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4711–4719.

[25] H. Yoo, S. Kang, R. Qiu, C. Xu, F. Wang, and H. Tong, "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*, 2025, pp. 2092–2101.

[26] C. Zhang, Z. Xue, G. Long, W. Zhang, and B. Yang, "Learning evolving preferences: A federated continual framework for user-centric recommendation," in *Proceedings of the ACM Web Conference 2026*, 2026, pp. 6609–6620.

[27] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 447–456.

[28] N. De Cao, W. Aziz, and I. Titov, "Editing factual knowledge in language models," in *Proceedings of the 2021 conference on empirical methods in natural language processing*, 2021, pp. 6491–6506.

[29] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, and C. Finn, "Memory-based model editing at scale," in *International Conference on Machine Learning*. PMLR, 2022, pp. 15 817–15 831.

[30] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," *Advances in neural information processing systems*, vol. 35, pp. 17 359–17 372, 2022.

[31] Z. Wu, A. Arora, Z. Wang, A. Geiger, D. Jurafsky, C. D. Manning, and C. Potts, "Reft: Representation finetuning for language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 63 908–63 962, 2024.

[32] P. Wang, Z. Li, N. Zhang, Z. Xu, Y. Yao, Y. Jiang, P. Xie, F. Huang, and H. Chen, "Wise: Rethinking the knowledge memory for lifelong model editing of large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 53 764–53 797, 2024.

[33] F. Li, X. Liu, J. Luo, G. Guo, J. Zhao, and X. Wang, "Lifelong sequential recommendation with adaptive subsequence compression and contextual fusion," in *Proceedings of the ACM Web Conference 2026*, 2026, pp. 5898–5907.

[34] X. Huang, J. Lian, Y. Lei, J. Yao, D. Lian, and X. Xie, "Recommender ai agent: Integrating large language models for interactive recommendations," *ACM Transactions on Information Systems*, vol. 43, no. 4, pp. 1–33, 2025.

[35] W. X. Zhao, J. Chen, P. Wang, Q. Gu, and J.-R. Wen, "Revisiting alternative experimental settings for evaluating top-n item recommendation algorithms," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2329–2332.

[36] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian *et al.*, "Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms," in *proceedings of the 30th acm international conference on information & knowledge management*, 2021, pp. 4653–4664.

[37] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 452–461.

[38] S. Choi, S. Kim, H. Kang, W. Kim, and C. Park, "Dynamic time-aware continual user representation learning," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025, pp. 740–749.

[39] J. Lim, W. Kweon, W. Kim, J. Kim, S. Choi, D. Kim, and H. Yu, "Federated continual recommendation," in *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, 2025, pp. 1798–1808.

[40] G. Lee, H. Yoo, J. Hwang, S. Kang, and H. Yu, "Capturing user interests from data streams for continual sequential recommendation," in *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, 2026, pp. 313–323.

A Additional Method Details and Proofs

This appendix provides additional definitions, implementation details, and proofs omitted from the main text. We first formalize the two-time-scale memory construction, then describe the value-aware teacher, the sparse low-rank editor, the inference-time edit calibration rule, and the training objective.

A.1 Two-Time-Scale Memory

Age normalization and memory initialization. The memory age feature used in Eq. (3) is defined as

$$\eta(\Delta t_u) = \frac{\log(1+\Delta t_u)}{\log(1+H)}, \tag{13}$$

where H is a fixed normalization horizon. The logarithmic transformation preserves the ordering of inactive users while avoiding extremely large values for long-inactive users.

At stage 0, no historical memory is available. After the warm-up stage, we initialize the working memory embedding and the anchor embedding as

$$m_u^{(0)} = \mathrm{Norm}(q_u^{(0)}), \qquad s_u^{(0)} = \mathrm{Norm}(q_u^{(0)}), \qquad t_u^{\mathrm{last}} = 0. \tag{14}$$

For users not observed in a later stage, the memory embeddings remain unchanged, while the age feature increases until they reappear.

State encoder. In implementation, the compact policy state is produced by a lightweight state encoder:

$$x_u^{(t)} = \mathrm{MLP}_x\left([q_u^{(t)}, s_u^{(t-1)}, d_u^{(t-1)}, \phi_\eta(\eta(\Delta t_u))]\right), \tag{15}$$

where $\phi_\eta(\cdot)$ projects the scalar age feature into a latent feature vector. The working memory is not separately concatenated because $m_u^{(t-1)} = s_u^{(t-1)} + d_u^{(t-1)}$. The conflict score is not used as an input to the policy state; it is used only to control the edit radius in Appendix A.3.

Assumption 1 (Noisy observation model for user embeddings)**.** *For a fixed user u, assume that the normalized current embedding at stage t can be written as*

$$\mathrm{Norm}(q_u^{(t)}) = \mu_u + \epsilon_t,$$

where μ_u is the user's underlying stable preference, and ϵ_t is zero-mean independent noise with covariance Σ.

Proposition 1 (Lower variance of the slow anchor)**.** *Under Assumption 1, consider an exponential moving average*

$$z^{(t)} = \beta z^{(t-1)} + (1-\beta)\mathrm{Norm}(q_u^{(t)}).$$

In the stationary regime, its covariance is

$$\mathrm{Cov}(z^{(t)}) = \frac{1-\beta}{1+\beta}\Sigma.$$

Therefore, if the anchor momentum is larger than the working-memory momentum, i.e., $\beta_s > \beta_m$, then the anchor embedding has lower stationary variance than the working memory embedding.

Proof. Unrolling the exponential moving average in the stationary regime gives

$$z^{(t)} = (1-\beta)\sum_{\ell=0}^{\infty}\beta^\ell(\mu_u + \epsilon_{t-\ell}).$$

Taking expectation gives

$$\mathbb{E}[z^{(t)}] = (1-\beta)\sum_{\ell=0}^{\infty}\beta^\ell\mu_u = \mu_u.$$

Since the noise terms are independent and have covariance Σ,

$$\mathrm{Cov}(z^{(t)}) = (1-\beta)^2\sum_{\ell=0}^{\infty}\beta^{2\ell}\Sigma = \frac{(1-\beta)^2}{1-\beta^2}\Sigma = \frac{1-\beta}{1+\beta}\Sigma.$$

The scalar factor $(1-\beta)/(1+\beta)$ is strictly decreasing in β for $\beta \in [0,1)$. Hence, $\beta_s > \beta_m$ implies that the slow anchor has lower stationary variance than the working memory. □

Assumption 2 (Stage-local state construction)**.** *At stage t, the current embedding $q_u^{(t)}$ is computed without using the target interaction being predicted. In addition, inherited memory embeddings $m_u^{(t-1)}$ and $s_u^{(t-1)}$ are frozen during the optimization of stage t and are written back only after the stage finishes.*

Proposition 2 (Stage-frozen memory avoids within-stage leakage)**.** *Under Assumption 2, the policy state $x_u^{(t)}$ in Eq.* (3) *does not contain future interactions within the same stage through memory updates.*

Proof. The state $x_u^{(t)}$ depends on the current embedding $q_u^{(t)}$, the anchor embedding $s_u^{(t-1)}$, the drift embedding $d_u^{(t-1)}$, and the memory age feature. By Assumption 2, $q_u^{(t)}$ excludes the target interaction. The memory embeddings are inherited from stage $t-1$ and are not updated during mini-batch optimization at stage t. Therefore, no interaction from a later mini-batch within the same stage can be written into the memory state used by an earlier or current training instance. Thus, $x_u^{(t)}$ avoids within-stage leakage through memory updates. □

A.2 Value-Aware Teacher Construction

Model-side potential outcome. For a fixed user-side state $x_u^{(t)}$, item embeddings, and memory action a, the candidate user embedding $h_{u,a}^{(t)}$ in Eq. (5) can be interpreted as a model-side potential outcome. The corresponding potential ranking utility is evaluated by $\ell_{u,j}^{(t)}(a)$ in Eq. (6). This comparison does not make a causal claim about real user behavior. It only asks what ranking utility the current model would obtain if a different memory intervention were applied to the same state.

Mean and uncertainty over sampled negatives. The empirical mean and standard deviation used in Eq. (7) are defined as

$$\hat{\mu}_u^{(t)}(a) = \frac{1}{M}\sum_{j=1}^{M} \ell_{u,j}^{(t)}(a), \qquad \hat{\sigma}_u^{(t)}(a) = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\left(\ell_{u,j}^{(t)}(a) - \hat{\mu}_u^{(t)}(a)\right)^2 + \varepsilon}. \tag{16}$$

The standard deviation term measures the instability of the estimated action utility across sampled negatives.

Intervention cost. The intervention cost in Eq. (7) is defined as

$$C_u^{(t)}(\text{retain}) = 0, \qquad C_u^{(t)}(\text{edit}) = c_e + \frac{\|\Delta_u^{(t)}\|_2^2}{\|m_u^{(t-1)}\|_2^2 + \varepsilon}, \qquad C_u^{(t)}(\text{decay}) = c_d \frac{(1-\gamma_u^{(t)})^2 \|d_u^{(t-1)}\|_2^2}{\|m_u^{(t-1)}\|_2^2 + \varepsilon}. \tag{17}$$

The edit cost discourages unnecessary memory rewriting, while the decay cost discourages excessive suppression of the drift embedding.

Edit advantage and budget projection. For a mini-batch $\mathcal{B}$, the edit advantage is

$$A_u^{(t)} = V_u^{(t)}(\text{edit}) - \max\{V_u^{(t)}(\text{retain}), V_u^{(t)}(\text{decay})\}. \tag{18}$$

Let $\mathcal{E}_{\mathcal{B}}$ be the set of top $\lfloor B_{\text{edit}}|\mathcal{B}|\rfloor$ users according to $A_u^{(t)}$. The projected value is

$$\bar{V}_u^{(t)}(a) = \begin{cases} V_u^{(t)}(a), & a \neq \text{edit}, \\ V_u^{(t)}(\text{edit}), & a = \text{edit},\ u \in \mathcal{E}_{\mathcal{B}}, \\ -\infty, & a = \text{edit},\ u \notin \mathcal{E}_{\mathcal{B}}. \end{cases} \tag{19}$$

This projection imposes the edit budget before teacher normalization and avoids adding a separate budget penalty to the training objective.

Proposition 3 (Budget projection as constrained edit assignment)**.** *For a fixed mini-batch $\mathcal{B}$ and fixed marginal values $V_u^{(t)}(a)$, selecting the top $\lfloor B_{\text{edit}}|\mathcal{B}|\rfloor$ users according to $A_u^{(t)}$ maximizes the total additional edit value under the constraint that at most $\lfloor B_{\text{edit}}|\mathcal{B}|\rfloor$ users can be assigned the edit action.*

Proof. For each user u, the best value among non-edit actions is

$$\max\{V_u^{(t)}(\text{retain}), V_u^{(t)}(\text{decay})\}.$$

Allowing edit instead adds exactly

$$A_u^{(t)} = V_u^{(t)}(\text{edit}) - \max\{V_u^{(t)}(\text{retain}), V_u^{(t)}(\text{decay})\}$$

additional value. Under the cardinality constraint that at most $\lfloor B_{\text{edit}}|\mathcal{B}|\rfloor$ users can be assigned edit, the objective is linear in the selected users. The optimal solution is therefore obtained by selecting the users with the largest edit advantages. This gives the projection rule in Eq. (19). □

A.3 Sparse Low-Rank Editor

The edit action directly modifies a user-specific memory embedding that will be reused in future stages. An unconstrained edit may improve the current training objective while corrupting stable preference for later recommendation. To avoid destructive editing, VAMIRec restricts the edit residual to be low-rank, sparse, and bounded.

Conflict score. The conflict score used to control the edit radius is

$$c_u^{(t)} = \left\| \text{Norm}(q_u^{(t)}) - \text{Norm}(s_u^{(t-1)} + d_u^{(t-1)}) \right\|_2. \tag{20}$$

This score is a deterministic mismatch measure between current evidence and inherited memory. It is not used as an input to the policy state.

Low-rank residual. The raw edit residual is produced by a bottleneck editor:

$$\Delta_{u,\text{raw}}^{(t)} = W_{\text{up}}\text{ReLU}(W_{\text{down}}x_u^{(t)}), \tag{21}$$

where $W_{\text{down}} \in \mathbb{R}^{r\times d_x}$, $W_{\text{up}} \in \mathbb{R}^{d\times r}$, and $r \ll d$. The bottleneck limits the capacity of the editor and prevents it from freely rewriting the whole memory embedding.

Straight-through sparse mask. Let $r_u^{(t)} = \text{MLP}_m(x_u^{(t)})$ be mask logits. The hard top-k operator $H_k(\cdot)$ keeps the largest k coordinates and sets the others to zero. We use a straight-through mask:

$$p_u^{(t)} = H_k(r_u^{(t)}) - \text{sg}(\sigma(r_u^{(t)})) + \sigma(r_u^{(t)}). \tag{22}$$

The masked residual is

$$\bar{\Delta}_u^{(t)} = p_u^{(t)} \odot \Delta_{u,\text{raw}}^{(t)}. \tag{23}$$

Conflict-dependent clipping. The edit radius is

$$\epsilon_u^{(t)} = \epsilon_{\min} + \epsilon_0\sigma(c_u^{(t)}), \tag{24}$$

so a larger current-memory conflict allows a larger but still bounded correction. The final edit residual is

$$\Delta_u^{(t)} = \text{Clip}(\bar{\Delta}_u^{(t)}, \epsilon_u^{(t)}) = \min\left(1, \frac{\epsilon_u^{(t)}}{\|\bar{\Delta}_u^{(t)}\|_2 + \varepsilon}\right)\bar{\Delta}_u^{(t)}. \tag{25}$$

Proposition 4 (Sparse and bounded edit)**.** *In the forward pass, the edit residual $\Delta_u^{(t)}$ changes at most k dimensions and satisfies*

$$\|\Delta_u^{(t)}\|_2 \le \epsilon_u^{(t)}.$$

Proof. In the forward pass, the straight-through mask in Eq. (22) has the same value as the hard top-k mask $H_k(r_u^{(t)})$. By definition, $H_k(r_u^{(t)})$ has at most k nonzero entries. Therefore, the masked residual $\bar{\Delta}_u^{(t)}$ in Eq. (23) can change at most k dimensions.

If $\|\bar{\Delta}_u^{(t)}\|_2 \le \epsilon_u^{(t)}$, the clipping operator in Eq. (25) returns $\bar{\Delta}_u^{(t)}$, so the norm bound holds. Otherwise, the operator rescales $\bar{\Delta}_u^{(t)}$ by

$$\frac{\epsilon_u^{(t)}}{\|\bar{\Delta}_u^{(t)}\|_2 + \varepsilon},$$

which makes the resulting norm no larger than $\epsilon_u^{(t)}$. Thus, the edit residual is both sparse and bounded. □

Proposition 5 (Low-rank edit subspace)**.** *Before sparsification, the raw edit residual $\Delta^{(t)}_{u,\mathrm{raw}}$ lies in the column space of W_{up} and therefore belongs to a subspace of dimension at most r.*

Proof. By Eq. (21),

$$\Delta^{(t)}_{u,\mathrm{raw}} = W_{\mathrm{up}} v^{(t)}_u, \qquad v^{(t)}_u = \psi(W_{\mathrm{down}} x^{(t)}_u) \in \mathbb{R}^r.$$

Thus, $\Delta^{(t)}_{u,\mathrm{raw}}$ is a linear combination of the columns of W_{up}. The dimension of the column space of W_{up} is at most r. Therefore, the raw edit residual belongs to a subspace of dimension at most r. □

Proposition 6 (Decay does not enlarge drift)**.** *Let the decay coefficient be*

$$\gamma^{(t)}_u = \gamma_{\min} + (1 - \gamma_{\min})\sigma(w_\gamma^\top x^{(t)}_u). \tag{26}$$

Then $\gamma^{(t)}_u \in [\gamma_{\min}, 1]$, and the decay candidate satisfies

$$\|\tilde{m}^{(t)}_{u,\mathrm{decay}} - s^{(t-1)}_u\|_2 \le \|d^{(t-1)}_u\|_2.$$

Proof. Since the sigmoid function outputs a value in $(0, 1)$, Eq. (26) gives

$$\gamma^{(t)}_u \in [\gamma_{\min}, 1].$$

For the decay action,

$$\tilde{m}^{(t)}_{u,\mathrm{decay}} = s^{(t-1)}_u + \gamma^{(t)}_u d^{(t-1)}_u.$$

Therefore,

$$\|\tilde{m}^{(t)}_{u,\mathrm{decay}} - s^{(t-1)}_u\|_2 = \gamma^{(t)}_u \|d^{(t-1)}_u\|_2 \le \|d^{(t-1)}_u\|_2.$$

Thus, decay can only preserve or attenuate the drift embedding and cannot enlarge it. □

A.4 Quantile Calibration and Expected Edit Rate

Let

$$r_\phi(x) = z_\phi(x, \mathrm{edit}) - \max\{z_\phi(x, \mathrm{retain}), z_\phi(x, \mathrm{decay})\}$$

be the edit advantage score. Let F_t denote the cumulative distribution function of $r_\phi(x)$ for stage-t user states.

Assumption 3 (Calibration distribution)**.** *Calibration states and test states at stage t are sampled from the same state distribution, and F_t is continuous.*

Proposition 7 (Expected edit-rate control)**.** *Under Assumption 3, if*

$$\theta_t = F_t^{-1}(1 - B_{\mathrm{edit}}),$$

then

$$\mathbb{E}_{x \sim F_t}[\mathbb{I}\{r_\phi(x) > \theta_t\}] = B_{\mathrm{edit}}.$$

If θ_t is estimated from n calibration states by the empirical quantile $\hat{\theta}_t$, then with probability at least $1 - \delta$,

$$\left|\mathbb{P}(r_\phi(x) > \hat{\theta}_t) - B_{\mathrm{edit}}\right| \le 2\sqrt{\frac{\log(2/\delta)}{2n}} + \zeta_t,$$

where ζ_t accounts for finite-sample quantile discretization and ties.

Proof. For the population threshold $\theta_t = F_t^{-1}(1 - B_{\mathrm{edit}})$ and continuous F_t, we have

$$\mathbb{P}(r_\phi(x) > \theta_t) = 1 - F_t(\theta_t) = B_{\mathrm{edit}}.$$

This proves the population edit-rate statement.

For the empirical threshold, let $\hat{F}_t$ be the empirical CDF computed from n calibration states. The Dvoretzky–Kiefer–Wolfowitz inequality gives

$$\sup_r |\hat{F}_t(r) - F_t(r)| \le \sqrt{\frac{\log(2/\delta)}{2n}}$$

with probability at least $1-\delta$. Since $\hat{\theta}_t$ is chosen as an empirical $(1-B_{\text{edit}})$-quantile, the empirical tail probability is within discretization error ζ_t of B_{edit}. Transferring this tail probability from $\hat{F}_t$ to F_t incurs at most twice the uniform CDF deviation. Thus,

$$\left|\mathbb{P}(r_\phi(x) > \hat{\theta}_t) - B_{\text{edit}}\right| \leq 2\sqrt{\frac{\log(2/\delta)}{2n}} + \zeta_t.$$

□

A.5 Learning Objective and Complexity

Straight-through action indicator. During training, the hard action used in the forward pass is combined with soft-policy gradients through

$$\tilde{\pi}_u^{(t)} = \text{onehot}(a_u^{(t)}) - \text{sg}(\pi_\phi^{(t)}) + \pi_\phi^{(t)}. \tag{27}$$

In the forward pass, $\tilde{\pi}_u^{(t)}$ equals the one-hot action indicator. In the backward pass, gradients are propagated through $\pi_\phi^{(t)}$.

Intervened memory. The intervened memory used for memory update is

$$\bar{m}_u^{(t)} = \sum_{a \in \mathcal{A}} \tilde{\pi}_u^{(t)}(a) \tilde{m}_{u,a}^{(t)}. \tag{28}$$

Recommendation loss. In our experiments, $\mathcal{L}_{\text{rec}}^{(t)}$ is instantiated as the BPR loss:

$$\mathcal{L}_{\text{rec}}^{(t)} = -\frac{1}{|\mathcal{B}|M} \sum_{u \in \mathcal{B}} \sum_{j=1}^{M} \log \sigma \left(h_u^{(t)\top} e_{i^+} - h_u^{(t)\top} e_{i_j^-} \right). \tag{29}$$

Complexity. Let d be the embedding dimension, r be the bottleneck rank, M be the number of sampled negatives, and $|\mathcal{A}| = 3$. The low-rank editor costs $O(rd)$ per user. The training-time teacher evaluates $|\mathcal{A}|$ candidate embeddings against one positive item and M sampled negatives, leading to additional training cost

$$O(|\mathcal{A}|Md).$$

Since $|\mathcal{A}|$ is constant, the overhead is linear in the number of sampled negatives. At inference time, the teacher and budget projection are not computed. The policy selects one action before all-item ranking, so the additional inference cost is independent of the number of candidate items except for standard dot-product scoring.

B Experimental Settings

In this section, we provide more details about the experimental settings, including dataset preprocessing, backbone models, baseline methods, and implementation details.

B.1 Dataset and Preprocessing

We evaluate VAMIRec on three real-world recommendation datasets: Amazon (CDs and Vinyls), Amazon (Video Games), and Gowalla. The two Amazon datasets are collected from online product review platforms, while Gowalla is a location-based social network dataset constructed from user check-in records. These datasets cover different recommendation scenarios and exhibit different levels of sparsity and temporal dynamics, making them suitable for evaluating continual recommendation methods.

For data preprocessing, we follow common implicit-feedback recommendation settings. Each observed user–item interaction is treated as positive feedback, and interactions are sorted chronologically to construct continual recommendation stages. Users and items with extremely sparse interactions are filtered out to ensure reliable training and evaluation. After preprocessing, each dataset is split into consecutive temporal stages. At each stage, the model is trained or updated using the interactions available up to the current stage, while future interactions are strictly excluded from training to avoid temporal leakage.

Table 3 presents the statistics of the datasets after preprocessing.

Table 3: Statistics of datasets after preprocessing. #Int denotes the number of interactions.

Dataset	#Users	#Items	#Int	#Stages	Avg. #Int/User
Amazon (CDs and Vinyls)	5,488	14,622	74,093	5	13.50
Amazon (Video Games)	9,826	16,172	93,471	5	9.51
Gowalla	61,302	636,558	1,716,274	5	28.00

B.2 Backbone and Baseline

We compare our proposed method with the following backbone models and baseline methods.

- **LightGCN** simplifies graph convolution for collaborative filtering by removing feature transformation and nonlinear activation, and only keeps neighborhood aggregation over the user–item interaction graph.
- **NGCF** adopts graph neural networks to propagate collaborative signals over the user–item interaction graph, enabling high-order user–item connectivity modeling.
- **UltraGCN** further simplifies graph-based recommendation by approximating infinite-layer graph convolution with constraint-based learning, improving both efficiency and effectiveness.
- **GRU4Rec** uses gated recurrent units to model chronological user behavior sequences and predicts the next item according to the learned sequential user representation.
- **SASRec** employs a self-attention mechanism to capture dependencies among historical interacted items and produces the current user representation for sequential recommendation.
- **BERT4Rec** adopts a bidirectional Transformer architecture with a Cloze-style objective, allowing the model to capture richer sequential dependencies from user interaction histories.
- **Fine-tune** incrementally updates the recommender using only the current-stage interactions. It represents a simple continual adaptation strategy, but may forget useful historical knowledge.
- **Full-Retrain** retrains the recommender from scratch at each stage using all interactions observed so far. It serves as a cumulative-data reference, but is computationally expensive in continual recommendation scenarios.
- **Replay** [18] stores a subset of historical interactions and mixes them with current-stage data during incremental training to alleviate catastrophic forgetting.
- **GraphSAIL** [22] incrementally updates GNN-based recommender models by distilling self-embedding information, local neighborhood structure, and global graph structure from the previous model. It serves as a representative graph-structure preservation baseline for continual recommendation.
- **LWC-KD** [23] introduces graph-structure-aware contrastive knowledge distillation for incremental recommender systems, encouraging the current model to preserve historical graph knowledge.
- **LWC-KD-PIW** [24] extends LWC-KD by introducing personalized imitation weights, so that users with different preference dynamics receive different strengths of historical distillation.
- **CCD** [9] proposes a continual collaborative distillation framework, where teacher and student recommenders are co-evolved to improve knowledge retention under continual updates.
- **PISA** [25] studies the stability–plasticity trade-off in continual recommender systems and explicitly improves plasticity for users with large preference shifts.

For the main comparison, we use LightGCN as the default backbone because several strong continual recommendation baselines are graph-based methods. To further evaluate the generality of VAMIRec, we additionally instantiate it on NGCF, UltraGCN, GRU4Rec, SASRec, and BERT4Rec in the backbone generalization study.

B.3 Implementation Details

We conduct all experiments on an Intel Core i7-11700KF platform with dual NVIDIA GeForce RTX 3090 (24GB) GPUs. The implementation is based on Python 3.8.19 and PyTorch 2.0.0. For all methods, the embedding size is set to 128 for a fair comparison. We use Adam as the optimizer, and the learning rate is selected from the validation set.

For graph-based backbones, the user–item interaction graph is constructed according to the interactions available at each continual stage. For sequential backbones, user interaction sequences are sorted chronologically, and the current-stage user representation is generated from the corresponding sequence encoder. For GRU4Rec, the number of GRU layers is set to 1. For SASRec and BERT4Rec, the number of self-attention layers is set to 2. The dropout rate of BERT4Rec is set to 0.6.

For continual training, all methods are evaluated under the same temporal split and all-item ranking protocol. At the first stage, no historical memory is available, and VAMIRec reduces to the backbone recommender. From the second stage onward, VAMIRec maintains user-side historical memory and applies retain, edit, or decay actions before ranking. During training, each positive interaction is paired with sampled negative items. During evaluation, all items that the user has not interacted with are treated as candidate items.

For VAMIRec, the default hyperparameters are selected according to validation performance. The imitation weight λ_{avi} controls the strength of action-policy distillation. The edit budget B_{edit} controls the proportion of users whose historical memory can be edited. The teacher temperature τ controls the smoothness of the action teacher distribution. The working-memory momentum β_m and the anchor momentum β_s control the update speeds of the working memory and slow anchor, respectively. We set $\beta_s > \beta_m$ so that the anchor evolves more slowly and serves as a stable preference reference.

Unless otherwise stated, we use the following default settings. The learning rate is set to 1×10^{-4}, the embedding size is set to 128, and the number of continual stages is set to 5. The default value of λ_{avi} is selected from $\{0.1, 0.3, 0.5, 0.7, 1.0\}$. The default value of B_{edit} is selected from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. The default value of τ is selected from $\{0.6, 0.8, 1.0, 1.2, 1.4\}$. The final selected values are determined by validation performance.

To reduce randomness, we first tune hyperparameters on the validation set with one random seed. After selecting the best hyperparameter configuration, we train each method with five different random seeds and report the average results. Recall@N and NDCG@N with $N \in \{10, 20\}$ are used as the evaluation metrics.

C More Experimental Results

C.1 Backbone Generalization Analysis

To evaluate whether VAMIRec is tied to a specific recommender architecture, we further instantiate it on both graph-based collaborative filtering backbones and sequential recommendation backbones. For graph-based recommendation, we use LightGCN, NGCF, and UltraGCN. For sequential recommendation, we include GRU4Rec, SASRec, and BERT4Rec. These backbones cover different representation learning paradigms, including graph propagation, recurrent sequence modeling, self-attentive sequence modeling, and bidirectional sequence encoding.

As shown in Table 4, VAMIRec consistently improves over the corresponding base graph backbones across datasets and metrics. This indicates that the proposed memory intervention mechanism is not merely an optimization trick for a single graph encoder. Instead, it can act as a backbone-level enhancement for continual recommendation, because it intervenes on user-side historical memory after the backbone has produced current-stage user representations.

The graph-based results also show that VAMIRec remains effective when combined with different collaborative filtering encoders. The gains on LightGCN are particularly clear, which is consistent with our default experimental setting. Similar trends on NGCF and UltraGCN further suggest that the proposed memory construction, action evaluation, and intervention policy are not specific to one graph architecture.

For sequential backbones, VAMIRec can be applied in the same way by treating the sequence encoder output as the current-stage user representation. Specifically, GRU4Rec, SASRec, and BERT4Rec

Table 4: Backbone generalization results on three benchmark datasets. We instantiate VAMIRec on both graph-based collaborative filtering backbones and sequential recommendation backbones to evaluate whether the proposed memory intervention mechanism provides consistent gains across different recommender architectures. For graph-based backbones, we compare the base model, the corresponding PISA-enhanced variant, and the VAMIRec-enhanced variant. For sequential backbones, we compare the base model and the VAMIRec-enhanced variant under the same continual recommendation protocol. The best results are highlighted in bold and the second-best results are underlined.

Backbone	Variant	Amazon (CDs and Vinyls)				Amazon (Video Games)				Gowalla			
		R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20
LightGCN	Base	0.71	0.46	1.40	0.66	0.88	0.55	1.44	0.73	2.15	1.57	3.07	1.86
	+PISA	0.89	0.62	1.58	0.83	1.55	1.03	2.20	1.22	2.90	2.16	4.21	2.47
	+VAMIRec	**1.06**	**0.66**	**1.81**	**0.88**	**1.76**	**1.11**	**2.68**	**1.39**	**3.13**	**2.35**	**4.29**	**2.65**
NGCF	Base	0.59	0.36	0.86	0.45	0.73	0.43	0.88	0.50	1.79	1.23	1.89	1.27
	+PISA	0.76	0.55	1.09	0.68	1.34	0.93	1.60	1.00	2.50	1.92	3.20	2.04
	+VAMIRec	**0.92**	**0.60**	**1.39**	**0.75**	**1.53**	**1.01**	**2.06**	**1.18**	**2.72**	**2.14**	**3.29**	**2.22**
UltraGCN	Base	0.45	0.25	0.93	0.38	0.56	0.30	0.96	0.42	1.36	0.85	2.04	1.07
	+PISA	0.63	0.42	1.06	0.54	1.15	0.70	1.49	0.80	2.13	1.45	2.86	1.62
	+VAMIRec	**0.80**	**0.46**	**1.23**	**0.59**	**1.33**	**0.77**	**1.82**	**0.93**	**2.36**	**1.64**	**2.92**	**1.75**
GRU4Rec	Base	0.68	0.42	1.22	0.58	1.05	0.66	1.78	0.87	2.04	1.46	3.20	1.86
	+VAMIRec	**0.88**	**0.55**	**1.48**	**0.72**	**1.39**	**0.88**	**2.14**	**1.09**	**2.55**	**1.82**	**3.78**	**2.18**
SASRec	Base	0.74	0.47	1.32	0.63	1.22	0.78	2.01	1.02	2.31	1.68	3.58	2.09
	+VAMIRec	**0.98**	**0.62**	**1.66**	**0.82**	**1.58**	**1.00**	**2.43**	**1.25**	**2.83**	**2.05**	**4.07**	**2.42**
BERT4Rec	Base	0.70	0.44	1.25	0.60	1.16	0.74	1.92	0.97	2.18	1.57	3.39	1.98
	+VAMIRec	**0.93**	**0.58**	**1.57**	**0.77**	**1.49**	**0.94**	**2.32**	**1.18**	**2.70**	**1.95**	**3.93**	**2.32**

produce the current representation from recent interaction sequences, while VAMIRec manages the inherited user memory through retain, edit, and decay actions. This design allows the proposed framework to be evaluated beyond graph-based recommendation and provides additional evidence for its architectural generality.

D Limitation

Two potential limitations of this work should be noted. First, VAMIRec involves several hyper-parameters, such as the imitation weight, edit budget, teacher temperature, and memory update momentum. Although they are selected according to validation performance, adapting them to different datasets or backbone recommenders may still require additional tuning effort. Second, the training-time action-value evaluation introduces extra computational overhead because retain, edit, and decay candidates are compared using sampled ranking feedback. This overhead does not affect all-item ranking at inference, but improving training efficiency would be beneficial for larger-scale deployment.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contributions and scope of the paper are included in the abstract and Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A limitation section is included in the Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: For each of our main theorems, we provide a detailed description of the modeling assumptions prior to the theorem. We provide complete proof in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We introduce the details of the experiment, such as the information on hardware and software, in the implementation detail section, *i.e.*, Appendix B.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have attached the data and code used in this paper in the supplementary material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the details of the experimental settings, such as the data split, optimizer, etc., in the experimental setting section (Section 4.1) in the main paper and the implementation detail section (Appendix B.3) in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the two-sided t-test with $p < 0.05$ results in the main experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the details of compute resources in Appendix B.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have made sure that our paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential positive impacts that our algorithm will bring in the Introduction section, *i.e.*, Section 1.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: There is no risk of misuse of the proposed method and the datasets used in the paper are open-sourced.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited the original paper or attached the link to the existing assets used in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have attached the introduction of how to run the code and the license in the code repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowd sourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We did not utilize LLM during the development of this research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.