TAMREC: TIME-AWARE AND MONOTONIC REINFORCEMENT LEARNING FOR ON-DEMAND EMBEDDING DIMENSIONS IN STREAMING RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In deep streaming recommender systems, fixed embedding dimensions are inefficient for streaming data, where user-item interactions and activity patterns continuously evolve. Existing dynamic adjustment methods, including reinforcement learning-based approaches, suffer from three critical issues: parameter inflation, catastrophic forgetting caused by shrink operations, and unstable rewards for cold-start. To address these challenges, this paper introduces TAMRec, employing a time-aware policy that decouples interaction frequency from capacity demand, and adopts a monotonic Maintain/Enlarge action space to eliminate shrink-induced forgetting. While monotonic expansion risks uncontrolled parameter growth, an adaptive reward mechanism encourages on-demand allocation: enlargement is favored only when it yields prediction performance above a moving baseline. A group-prior fallback further stabilizes learning for cold-start entities with sparse feedback. Theoretically, we prove that the policy converges to a local optimum with near-zero catastrophic forgetting, on-demand optimality, and bounded storage growth under mild assumptions. Experiments on five benchmark datasets demonstrate that TAMRec delivers competitive or better recommendation performance with compact embeddings and less forgetting in streaming settings. [1]

1 INTRODUCTION

Embedding tables are a major resource bottleneck in deep recommender systems Gheewala et al. (2025); Ibrahim et al. (2025). Fixed dimensions can over-allocate capacity to sparse or dormant users while limiting entities whose behavior becomes more complex. Online training updates embedding values and network weights, but does not change the capacity allocated to each entity Qu et al. (2024); He et al. (2023). Our Fixed-128 baseline receives the same online updates as TAMRec, isolating capacity allocation from weight adaptation.

This distinction matters in a changing stream. A dormant entity retains its allocated dimensions even when few observations support them, while additional gradient updates cannot increase the capacity of an entity whose interests become more diverse. The question is therefore when to allocate additional dimensions, rather than whether to update existing parameters. Temporal recency provides information that cumulative frequency alone cannot express: entities with similar interaction counts may have very different recent activity patterns.

Static mixed-dimension methods, such as DARTS Liu et al. (2019) and AutoEmb Zhao et al. (2021), tailor capacity but are difficult to adapt to evolving streams. Streaming methods, including ESAPN Liu et al. (2020) and SCALL Qu et al. (2024), adjust dimensions dynamically. However, cumulative-frequency-driven growth can inflate storage He et al. (2023); Luo et al. (2024), while shrink or mask operations may discard learned representations when past behaviors recur Song et al. (2023); Zhou & Li (2026); Yoo et al. (2025a). Sparse feedback also makes reinforcement learning (RL) rewards unstable for cold-start entities Ji et al. (2021); Singh & Singh (2024).

We propose TAMRec, a **T**ime-**A**ware and **M**onotonic RL framework for streaming **Rec**ommendation. Temporal intervals complement cumulative frequency: repeated short-term interactions need not

[1] We release the datasets and core codes at: https://anonymous.4open.science/r/STARLA-8887YZ/

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

justify expansion, whereas changing interaction patterns may indicate additional capacity demand. A policy with multi-head attention Vaswani et al. (2017) selects only *Maintain* or *Enlarge*, avoiding dimension removal. An adaptive reward favors prediction improvements over a moving baseline, with a group-prior fallback for sparse entities. Allocation is bounded by a predefined maximum dimension; diminishing returns motivate stopping below that limit Guo et al. (2024); He et al. (2025); Killingback et al. (2026).

Our contributions are threefold:

- **Time-aware on-demand allocation.** A policy combining frequency, current capacity, and temporal recency adapts embedding dimensions using performance feedback rather than cumulative frequency alone.
- **Monotonic dimension adjustment.** Restricting actions to Maintain and Enlarge preserves allocated dimensions and avoids their removal by shrink operations.
- **Sparse-feedback reward stabilization.** Group-prior fallbacks and time-decayed historical losses provide an adaptive reference for evaluating capacity decisions.

Experiments evaluate recommendation quality, storage, and forgetting on five datasets with two backbones.

2 Related Work

Streaming recommendation. NCF, DeepFM, LightGCN, SASRec, and BERT4Rec improve interaction modeling while using fixed-capacity embeddings He et al. (2017); Guo et al. (2017); He et al. (2020); Kang & McAuley (2018); Sun et al. (2019). Embedding storage dominates many large-scale systems Naumov et al. (2019). Streaming and continual approaches adapt parameters or replay historical data Wang et al. (2018); Guo et al. (2019); Vitter (1985); Dai et al. (2024); Yoo et al. (2025b); dynamic capacity allocation addresses a complementary problem.

Dynamic embedding dimensions. DARTS and AutoEmb search mixed dimensions Liu et al. (2019); Zhao et al. (2021); ESAPN, DESS, SCALL, and MultiEmb introduce online allocation, masking, or pruning Liu et al. (2020); He et al. (2023); Qu et al. (2024); Yang et al. (2025). Frequency-driven expansion can waste capacity, while dimension removal risks forgetting Luo et al. (2024); Song et al. (2023); Yoo et al. (2025a). TAMRec combines temporal signals with monotonic adjustment and performance-based rewards.

RL for recommendation. RL recommenders usually select items or slates using user feedback Zheng et al. (2018); Chen et al. (2019); Afsar et al. (2022); Lin et al. (2023), including approaches incorporating language models Wang et al. (2024; 2025); Zhang et al. (2025). TAMRec instead uses RL to control embedding capacity, making it complementary to these recommendation policies.

3 Methodology

TAMRec addresses parameter inflation, catastrophic forgetting, and cold-start reward instability by unifying time-aware state modeling, monotonic shrink-free actions, and group-prior stabilized rewards. The framework comprises two core modules: a streaming recommendation backbone and a time-aware monotonic RL controller. Figure 1 shows the framework structure.

3.1 Streaming Recommendation Backbone

We target streaming recommendation under the inference-then-training protocol: interactions arrive sequentially, the model predicts user preference, evaluates performance, and updates parameters immediately without replay.

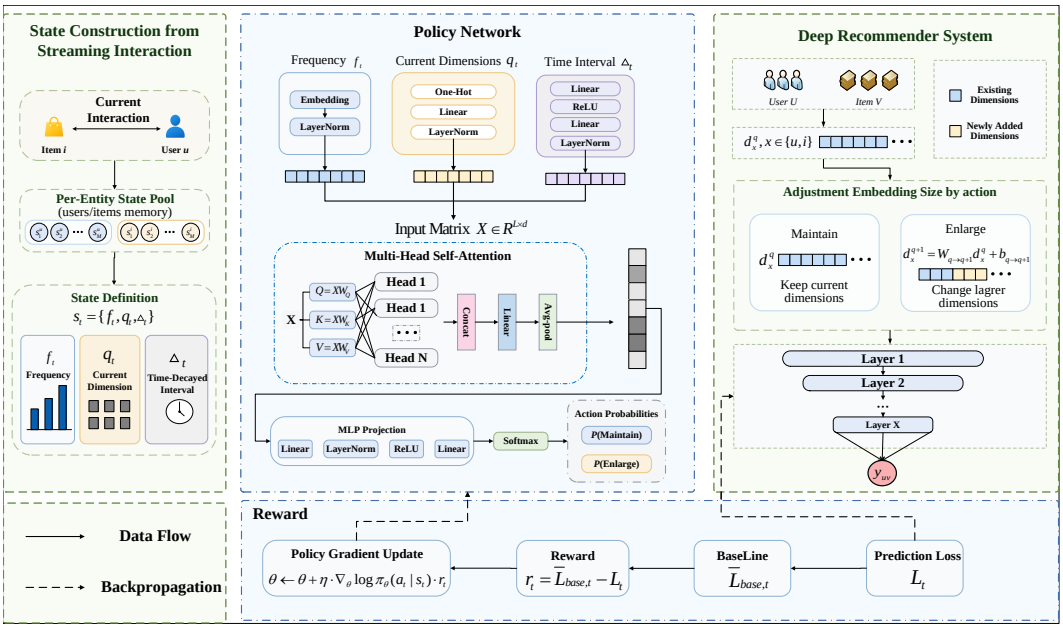


Figure 1: Overview of TAMRec. The framework illustrates the full streaming recommendation pipeline, including entity-level state construction, the time-aware policy network, monotonic embedding dimension adjustment, the deep recommender system, and reward-based policy update.

The backbone takes dynamically sized user/item embeddings as input and outputs interaction scores. To align embeddings across dimensions, we use hierarchical projection:

$$e^{(j+1)} = \mathrm{Tanh}\left(\mathrm{BN}\left(W_{j \to j+1} e^{(j)} + b_{j \to j+1}\right)\right), \tag{1}$$

where $e^{(j)} \in \mathbb{R}^{d_j}$ denotes the embedding at the j-th candidate dimension, $e^{(j+1)} \in \mathbb{R}^{d_{j+1}}$ is the projected embedding at the next larger dimension, and $W_{j \to j+1} \in \mathbb{R}^{d_{j+1} \times d_j}$ and $b_{j \to j+1} \in \mathbb{R}^{d_{j+1}}$ are the learnable projection matrix and bias term, respectively. Here, $\mathrm{BN}(\cdot)$ denotes batch normalization and $\mathrm{Tanh}(\cdot)$ denotes the nonlinear activation.

All embeddings are projected to the maximum dimension $d_{\max}$, concatenated, and fed into an MLP predictor. The model is optimized by cross-entropy loss for explicit feedback or ranking loss for implicit feedback.

3.2 Time-Aware Monotonic RL Controller

The RL controller learns to assign embedding dimensions on demand, expanding only when performance gains justify higher capacity, rather than following cumulative frequency.

3.2.1 State Definition

The policy state integrates three complementary entity-level signals to decouple activity frequency from genuine capacity demand:

$$s_t = [f_t, q_t, \Delta_t], \tag{2}$$

where f_t denotes the cumulative interaction frequency up to t, q_t represents the current embedding dimension index (reflecting the current capacity level), and Δ_t is the time-decayed interval feature.

The time-decayed interval is computed as:

$$\Delta_t = \begin{cases} \epsilon, & \text{if the current entity is newly observed,} \\ \exp\left(-\frac{\lambda(\tau_t - \tau_{\text{last}})}{T_{\text{day}}}\right) + \epsilon, & \text{otherwise,} \end{cases} \tag{3}$$

where τ_t is the current timestamp, τ_{last} is the last observed timestamp, $T_{\text{day}} = 86400$ normalizes the interval to days, λ controls temporal sensitivity, and ϵ ensures numerical stability.

A larger Δ_t value indicates dense recent events with short time intervals; a smaller value implies a long inactive period and sparse recent events.

3.2.2 MONOTONIC ACTION SPACE

The action space is constrained to Maintain/Enlarge; Shrink is explicitly excluded to guarantee monotonic non-decreasing dimensions and near-zero catastrophic forgetting:

$$a_t \sim \pi_\theta(\cdot \mid s_t), \quad a_t \in \{\text{Maintain}, \text{Enlarge}\}, \tag{4}$$

where θ denotes the policy parameters.

This shrink-free design is motivated by two critical observations:

1. Catastrophic forgetting induced by dimension reduction is the main cause of performance degradation in streaming recommendation, which is eliminated by prohibiting any shrink operation.

2. Embedding performance exhibits sharply diminishing marginal returns Killingback et al. (2026); He et al. (2025). In recommender systems, dimensions beyond 128–256 yield negligible gains and even risk overfitting or embedding collapse Guo et al. (2024). Thus, an enlarge-only policy does not lead to uncontrolled storage growth, as the policy will naturally stop expanding once the entity-specific optimal dimension is reached.

3.2.3 ADAPTIVE REWARD WITH GROUP-PRIOR FALLBACK

We design an adaptive baseline loss $\bar{L}_{\text{base},t}$ to stabilize rewards, especially for cold-start and sparse entities:

$$\bar{L}_{\text{base},t} = \begin{cases} \frac{1}{|G_{\text{peer}}|} \sum_{m \in G_{\text{peer}}} \bar{L}_m, & N_x < K, \\ \frac{\sum_{j=1}^{T_x} \exp(-\beta \Delta\tau_{x,j}^{\text{hist}}) L_{x,j}^{\text{hist}}}{\sum_{j=1}^{T_x} \exp(-\beta \Delta\tau_{x,j}^{\text{hist}})}, & \text{otherwise}, \end{cases} \tag{5}$$

where N_x is the interaction count of entity x, K is the sparsity threshold, G_{peer} is the peer group with the same frequency, and β controls historical loss decay.

Sparse entities ($N_x < K$) use a group prior (same-frequency peers) for robust fallback. Dense entities use a time-decayed historical loss baseline.

The immediate reward measures performance improvement over the baseline:

$$r(s_t, a_t) = \bar{L}_{\text{base},t} - L_t, \tag{6}$$

where L_t is the current prediction loss. A positive reward indicates that the action improves performance over the baseline.

Positive reward validates beneficial dimension enlargement; otherwise, Maintain is favored to suppress redundant growth.

3.2.4 POLICY OPTIMIZATION

The time-aware policy network (with multi-head attention) disentangles frequency, capacity, and temporal signals. The policy is optimized via policy gradient ascent, using the adaptive baseline $b(s_t) = \bar{L}_{\text{base},t}$ to reduce gradient variance.

We aim to maximize the expected cumulative reward $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} r(s_t, a_t)\right]$ (or average reward per step), and we denote its gradient by:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot r(s_t, a_t)\right]. \tag{7}$$

Its estimator with baseline is:

$$\hat{\nabla}_\theta J(\theta) = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \big(r(s_t, a_t) - b(s_t)\big). \tag{8}$$

The objective maximizes expected cumulative reward for long-term on-demand optimality. We optimize the policy parameters θ via policy gradient ascent, with the update rule:

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta J(\theta_k), \tag{9}$$

where α_k is the step size at iteration k.

3.3 Theoretical Analysis

Appendix A develops six results under the assumptions stated for each theorem. They connect the controller's optimization, reward design, and allocation behavior:

- **Optimization and reward estimation.** The convergence analysis considers a finite dimension set, Lipschitz dynamics, and stochastic-approximation step sizes. An action-independent baseline preserves the expected policy gradient, while the variance analysis characterizes the baseline that minimizes conditional estimator variance.
- **Retention and allocation.** Monotonic adjustment preserves the set of allocated dimensions. The forgetting analysis studies the resulting retention under its stated assumptions; the allocation analysis relates enlargement to marginal accuracy gains and storage cost. Preserving dimensions specifically removes shrink-induced information loss, rather than by itself excluding every source of forgetting during weight updates.
- **Sparse feedback and bounded growth.** The group-prior analysis examines variance reduction from pooling peer losses when individual histories are short. Under the reward and saturation assumptions, the storage analysis describes conditions under which average dimension remains below the maximum.

The appendix also gives storage and per-batch computational complexity. These conditional results motivate the design; the experiments below separately assess its accuracy, capacity usage, and temporal-task behavior.

4 Experiments

We evaluate TAMRec on five real-world datasets under a chronological streaming protocol. The experiments cover eight aspects: overall recommendation performance (RQ1), embedding space efficiency (RQ2), dimension allocation with respect to frequency and temporal intervals (RQ3), component ablations (RQ4), cold-start and long-tail robustness (RQ5), catastrophic forgetting under shrink-based adjustment (RQ6), computational efficiency (RQ7), and hyperparameter sensitivity (RQ8).

4.1 Experimental Setup

We compare Fixed-128, DARTS Liu et al. (2019), AutoEmb Zhao et al. (2021), ESAPN Liu et al. (2020), SCALL Qu et al. (2024), and MultiEmb Yang et al. (2025). Fixed-128 updates both embeddings and network weights online under the same inference-then-training protocol as TAMRec. SCALL uses memory budget $c = 32$; MultiEmb splits MovieLens by genre and Netflix by time.

We use MovieLens-20M, MovieLens-Latest, and Netflix for explicit feedback, and Amazon Beauty and LastFM-1K for implicit feedback. Interactions are ordered chronologically. Every method is evaluated with MLP and NeuMF backbones, the latter combining matrix factorization and MLP branches. We use candidate dimensions $\{4, 8, 16, 32, 64, 128\}$ and Adam, reporting accuracy/CE and HR/NDCG for explicit and implicit feedback, respectively. Appendix B provides implementation and evaluation details.

Under the streaming protocol, each incoming batch is first used for prediction and evaluation, then for parameter updates. The warm-up stage freezes the policy while the recommender accumulates loss statistics; subsequent training updates both modules. This order prevents evaluation on a batch after fitting that same batch. Using two interaction backbones tests whether the observed allocation benefits persist beyond the default MLP, while the fixed-capacity baseline controls for ordinary online weight adaptation.

4.2 RQ1: Overall Performance

Tables 1 and 2 report results for both backbones. TAMRec has the highest accuracy on all three explicit-feedback datasets. On Netflix with MLP, accuracy increases from the best baseline's 0.4741 to 0.4859. CE improvements are not universal: MultiEmb has lower CE on MovieLens-Latest with MLP, and ESAPN has lower CE on Netflix with NeuMF (1.1626 vs. 1.1666).

Table 1: Overall performance comparison under the streaming evaluation protocol. Higher accuracy and lower cross-entropy (CE) loss indicate better performance. Best results are highlighted in bold.

Backbone	Method	MovieLens-20M		MovieLens-Latest		Netflix	
		Accuracy ↑	CE Loss ↓	Accuracy ↑	CE Loss ↓	Accuracy ↑	CE Loss ↓
MLP	FIXED	0.4479	1.2651	0.4284	1.3039	0.4254	1.2939
	DARTS	0.4497	1.2599	0.4355	1.2852	0.4548	1.2422
	ESAPN	0.4535	1.2487	0.4423	1.2715	0.4149	1.3025
	AutoEmb	0.4522	1.2540	0.4327	1.2852	0.4566	1.2343
	SCALL(c=32)	0.4435	1.2633	0.4438	1.2773	0.4707	1.2042
	MultiEmb	0.4501	1.2493	0.4441	**1.2633**	0.4741	1.1917
	TAMRec*	**0.4539**	**1.2477**	**0.4445**	1.2721	**0.4859**	**1.1551**
NeuMF	FIXED	0.4500	1.2585	0.4357	1.2854	0.4489	1.2534
	DARTS	0.4508	1.2568	0.4375	1.2823	0.4605	1.2313
	ESAPN	0.4616	1.2282	0.4460	1.2596	0.4898	**1.1626**
	AutoEmb	0.4563	1.2459	0.4351	1.2901	0.4562	1.2348
	SCALL(c=32)	0.4551	1.2440	0.4452	1.2610	0.4874	1.1685
	MultiEmb	0.4541	1.2496	0.4426	1.2673	0.4847	1.1751
	TAMRec*	**0.4653**	**1.2003**	**0.4480**	**1.2544**	**0.4910**	1.1666

Table 2: Overall performance comparison under the streaming evaluation protocol. Higher values indicate better performance. Best results are highlighted in bold.

Backbone	Method	Amazon Beauty				LastFM-1K			
		HR@10	HR@20	NDCG@10	NDCG@20	HR@10	HR@20	NDCG@10	NDCG@20
MLP	FIXED	0.1123	0.1424	0.0619	0.0694	0.1474	0.2304	0.0850	0.1100
	DARTS	0.1289	0.1571	0.0757	0.0828	0.1382	0.2320	0.0750	0.1000
	ESAPN	0.1267	0.1373	0.0652	0.0718	0.1509	0.2380	0.0850	0.1100
	AutoEmb	0.1161	0.1244	0.0605	0.0696	0.1420	0.2200	0.0800	0.1050
	SCALL(c=32)	0.1251	0.1583	0.0720	0.0815	0.1650	0.2455	0.0952	0.1204
	MultiEmb	0.1210	0.1311	**0.0794**	**0.0831**	0.1784	0.2553	0.1050	**0.1300**
	TAMRec*	**0.1316**	**0.1700**	0.0713	0.0810	**0.1892**	**0.2621**	**0.1101**	0.1285
NeuMF	FIXED	0.1206	0.1681	0.0662	0.0782	0.2760	0.3669	0.1608	0.1837
	DARTS	0.1247	0.1649	0.0603	0.0706	0.1249	0.1862	0.0671	0.0825
	ESAPN	0.0856	0.1234	0.0486	0.0581	0.1976	0.2613	0.1191	0.1351
	AutoEmb	0.0276	0.0427	0.0132	0.0170	0.0059	0.0094	0.0032	0.0041
	SCALL(c=32)	0.1008	0.1374	0.0548	0.0642	0.2530	0.3248	0.1568	0.1748
	MultiEmb	0.0950	0.1310	0.0515	0.0605	0.2400	0.3100	0.1480	0.1650
	TAMRec*	**0.1262**	**0.1705**	**0.0678**	**0.0797**	**0.3085**	**0.3883**	**0.1944**	**0.2145**

On implicit-feedback datasets, TAMRec has the highest HR@10 and HR@20 with both backbones. Some MLP NDCG metrics favor MultiEmb, whereas TAMRec leads all reported NeuMF ranking metrics. On LastFM-1K with NeuMF, HR@10 increases from 0.2760 to 0.3085 (11.8%). On Amazon Beauty, it exceeds Fixed-128 in HR@10 (0.1262 vs. 0.1206) and NDCG@10 (0.0678 vs. 0.0662), using 3.58% and 7.65% of its user and item embedding parameters. These results support the benefit of adaptive capacity under the same online training schedule.

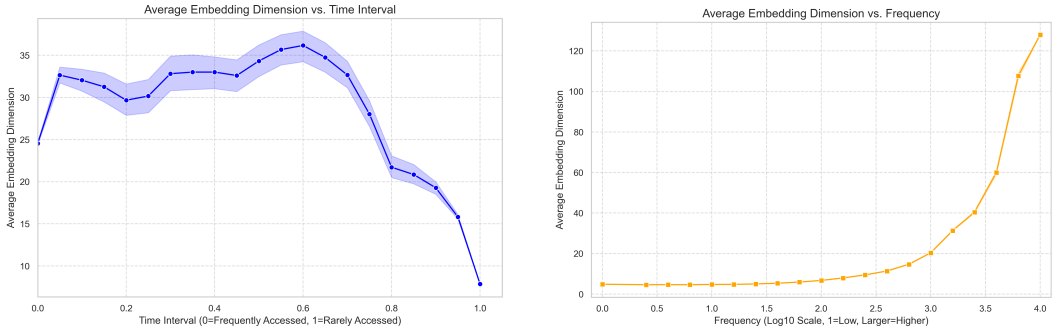
4.3 RQ2: Space Efficiency Analysis

Table 3 reports embedding ratios relative to Fixed-128. With MLP, TAMRec uses 4.09%, 14.01%, and 8.92% of full user capacity on Amazon Beauty, MovieLens-20M, and MovieLens-Latest, respectively. It allocates more on Netflix (55.79%) and LastFM-1K (99.00%), reflecting dataset-dependent accuracy–storage tradeoffs.

The same qualitative pattern holds with NeuMF: Amazon Beauty uses 3.58% user and 7.65% item capacity, whereas LastFM-1K uses 91.02% user capacity. TAMRec has the smallest item embeddings on both MovieLens datasets. It is not uniformly the most compact method: on Netflix, SCALL achieves comparable NeuMF accuracy with about one third of TAMRec's capacity. Thus, allocation depends on both dataset and backbone, and explicit budgets remain a useful complementary constraint.

Table 3: Embedding efficiency comparison across different datasets. Lower ratio indicates more compact representations. (For MultiEmb, the reported ratios correspond to domain-specific embeddings instead of user/item embeddings.)

Backbone	Method	Amazon Beauty		LastFM-1K		MovieLens-20M		MovieLens-Latest		Netflix	
		User ↓	Item ↓	User ↓	Item ↓	User ↓	Item ↓	User ↓	Item ↓	User ↓	Item ↓
MLP	FIXED	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	DARTS	0.0621	0.1434	0.8534	0.6517	0.2010	0.3038	0.1628	0.2644	0.3360	0.4386
	ESAPN	**0.0380**	0.1545	0.7664	0.6631	0.2203	0.2645	0.1684	**0.1792**	0.3390	0.9994
	AutoEmb	0.0763	0.1210	0.8200	0.6000	0.1426	0.5720	0.1017	0.1804	0.4597	0.9998
	SCALL(c=32)	0.0450	0.1204	**0.1846**	**0.2327**	0.2449	**0.2438**	0.1846	0.2446	**0.2350**	**0.2350**
	MultiEmb	0.4949	0.2481	0.4200	0.4238	0.2585	0.2585	0.2592	0.2592	0.3054	0.3054
	TAMRec*	0.0409	**0.1168**	0.9900	0.5548	**0.1401**	**0.1683**	**0.0892**	0.2185	0.5579	0.6251
NeuMF	FIXED	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	DARTS	0.0395	0.1137	0.4367	0.2189	**0.1995**	0.0661	0.1650	0.2200	**0.1848**	0.2653
	ESAPN	**0.0354**	0.1083	0.9861	0.6985	0.2193	0.1575	0.1688	0.4380	0.7943	0.9997
	AutoEmb	0.0720	**0.0591**	**0.2151**	**0.1352**	0.2407	0.2233	**0.1450**	0.2553	0.4211	0.9891
	SCALL(c=32)	0.0362	0.1021	0.2440	0.2375	0.2420	0.2360	0.2460	0.2410	0.2430	**0.2450**
	MultiEmb	0.0356	0.0914	0.9425	0.5620	0.2742	0.1042	0.2035	0.3017	0.8612	0.9952
	TAMRec*	0.0358	0.0765	0.9102	0.4342	0.3311	**0.0515**	0.2419	**0.1698**	0.7021	0.7846



(a) Time-Decayed Interval Feature

(b) Cumulative Frequency

Figure 2: Embedding dimension trends w.r.t. time-decayed interval feature (left) and cumulative frequency (right).

4.4 RQ3: Frequency and Temporal Allocation Patterns

Figure 2 shows a non-monotonic relationship between allocated dimension and the time-decayed feature. Capacity is low near 1 (frequent recent interactions), peaks around 0.6, and is lower near 0 (prolonged inactivity). Allocated dimensions also increase with cumulative frequency. These patterns are consistent with allocation that uses temporal information alongside frequency, rather than expanding solely with interaction count.

4.5 RQ4: Ablation Study

Table 4 summarizes the component comparisons.

Table 4: Selected component ablations. Complete metrics and embedding ratios for all five datasets are in Appendix H.

Method	Netflix Accuracy ↑	Amazon Beauty HR@10 ↑	LastFM-1K HR@10 ↑
TAMRec	**0.4859**	**0.1316**	**0.1892**
w/o Time-Aware	0.4201	0.1043	0.1569
w/o Attention	0.4833	0.0962	0.1595
w/o Group Prior	0.4813	0.1042	0.1623

Removing time-aware features lowers Netflix accuracy from 0.4859 to 0.4201 and increases the MovieLens-20M user ratio from 0.1401 to 0.2286. Removing attention lowers MovieLens-20M accuracy from 0.4540 to 0.4517, while removing group priors raises its user ratio to 0.2936. On Amazon Beauty, HR@10 falls from 0.1316 to 0.1043, 0.0962, and 0.1042 without time awareness, attention, and group priors, respectively. Full accuracy, ranking, and storage results appear in Appendix H, Tables 13–14.

The storage measurements help interpret these accuracy changes. Without group priors, the Netflix user ratio increases from 0.5579 to 0.7807 even as accuracy falls from 0.4859 to 0.4813. On LastFM-1K, removing time awareness raises the item ratio from 0.5548 to 0.7363 while reducing HR@10 from 0.1892 to 0.1569. These cases show that allocating more dimensions does not necessarily recover the benefit of the complete controller. Conversely, removing attention can reduce some capacity ratios while also reducing accuracy, so storage savings should be assessed jointly with predictive performance.

4.6 RQ5–RQ8: Robustness and Efficiency

Cold-start and long-tail users (RQ5). Figure 3 compares performance by user activity. For users with 1–5 interactions, TAMRec reaches accuracy 0.3421, compared with 0.3352 for SCALL and 0.3287 for ESAPN. It also retains an advantage for users with 6–20 interactions, reaching 0.3988. Differences narrow for more active users (Appendix F). These observations are consistent with the motivation for group-prior rewards: individual loss histories are least informative at the beginning of a stream.

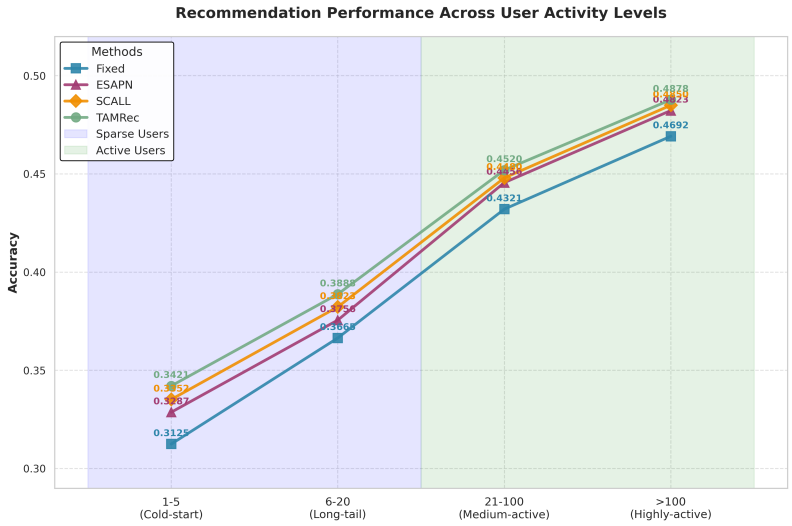


Figure 3: Recommendation accuracy across user activity levels. TAMRec shows clear advantages on cold-start and long-tail users while maintaining competitive performance on highly active users.

Temporal-task retention (RQ6). We divide the stream into five chronological tasks and evaluate backward transfer (BWT) and the forgetting measure (FM); their definitions appear in Appendix E. Positive BWT indicates improvement on past tasks after subsequent training, whereas lower FM indicates less deterioration from their best historical performance. Table 5 reports the results.

SCALL has negative BWT and higher FM, whereas Fixed-128, ESAPN, and TAMRec have near-zero FM. TAMRec attains final average HR@10 of 0.1142 and BWT of 0.0704, compared with 0.1001 and 0.0536 for ESAPN. Fixed-128 also avoids substantial forgetting, but has lower final performance. Thus, the comparison distinguishes retention from predictive quality: avoiding shrinkage preserves capacity, while the time-aware controller determines how that capacity is allocated.

Computational efficiency (RQ7). On Amazon Beauty, TAMRec takes 16.61 minutes versus 16.85 for Fixed-128, with average user and item dimensions of 4.5 and 16.3. The policy adds

Table 5: Catastrophic Forgetting Evaluation using Continual Learning metrics. Higher BWT is better; lower FM is better.

Method	HR@10			NDCG@10		
	Final Avg ↑	BWT ↑	FM ↓	Final Avg ↑	BWT ↑	FM ↓
Fixed-128 (Static)	0.0970	0.0409	0.0000	0.0392	0.0122	0.0002
SCALL (Shrink-based)	0.0753	-0.0001	0.0054	0.0301	-0.0002	0.0025
ESAPN (Monotonic)	0.1001	0.0536	0.0000	0.0512	0.0285	0.0000
TAMRec (Ours)	**0.1142**	**0.0704**	**0.0000**	**0.0551**	**0.0371**	**0.0000**

computation, but compact embeddings reduce the recommender's cost; the total measured runtime remains comparable. Appendix C reports the shared hardware configuration and the per-batch breakdown, including controller updates and embedding transfers.

The per-batch breakdown makes this tradeoff explicit. Controller updates contribute 11.62 ms and policy lookup plus embedding transfer contributes 1.45 ms, while the recommender backward pass remains close to Fixed-128 (18.26 vs. 18.60 ms). These measurements distinguish the additional policy cost from the total streaming runtime. They support the feasibility of adaptive allocation in the tested setting, but do not establish a universal speedup: the balance depends on the resulting dimensions, workload, and implementation.

Hyperparameter sensitivity (RQ8). We vary the reward window K, temporal coefficient λ, and reward-decay coefficient β separately on Amazon Beauty. Across the tested settings, HR@10 ranges from 0.1291 to 0.1322. Increasing K from 1 to 10 reduces the item ratio from 0.1744 to 0.0890. At $K = 10$, the ratio is 27.0% below the default and HR@10 differs by 0.0008. Thus, window size affects storage more strongly than ranking accuracy in this experiment. The decay-coefficient sweeps show smaller allocation changes; the complete ranges and results appear in Appendix D.

The sensitivity sweep uses $K \in \{1, 3, 5, 10\}$, $\lambda \in \{1, 2, 5, 10, 20\}$, and $\beta \in \{0, 0.3, 0.5, 0.7, 1.0\}$, changing one parameter at a time while retaining the other defaults. The small accuracy variation therefore describes local robustness over these tested ranges, rather than joint optimization of all hyperparameters. The larger change in storage with K suggests that reward-history aggregation is a practical allocation control. By contrast, the relatively small changes across the decay sweeps do not justify selecting a different coefficient from isolated improvements in one metric.

Scope and limitations. The results concern user/item embedding allocation under chronological streaming evaluation. The dimension set is discrete and capped at 128, so an entity's preferred capacity may fall between available levels. Moreover, although temporal recency informs allocation, the experiments do not independently evaluate preference-drift detection. Industrial streams, larger dimension ranges, and different recommendation architectures remain important tests of generality. Appendix G discusses these limits and broader implications.

5 Conclusions

TAMRec combines time-aware state modeling, monotonic dimension adjustment, and group-prior rewards for streaming recommendation. Across five datasets and two backbones, it achieves competitive recommendation quality with adaptive embedding storage; the temporal-task evaluation supports reduced forgetting relative to shrink-based adjustment. The study uses a maximum dimension of 128 and collaborative-filtering datasets. Larger capacities, industrial streams, and other architectures require further evaluation.

A practical next step is to combine temporal allocation with an explicit storage budget, motivated by SCALL's compact Netflix configuration. Such an extension should evaluate accuracy, memory, runtime, and retention together: improvements in one measure alone do not establish a better streaming system. The present results provide a basis for examining these tradeoffs across allocation policies and recommendation backbones.

REFERENCES

M. Mehdi Afsar, Trafford Crump, and Behrouz H. Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, pp. 532–547, 2018.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 456–464, 2019.

Binyang Dai, Xiao Hu, and Ke Yi. Reservoir sampling over joins. *Proceedings of the ACM on Management of Data*, 2(3):1–26, 2024. doi: 10.1145/3654921.

S. Gheewala, S. Xu, and S. Yeom. In-depth survey: Deep learning in recommender systems—exploring prediction and ranking models, datasets, feature analysis, and emerging trends. *Neural Computing and Applications*, pp. 10875–10947, 2025. doi: 10.1007/s00521-024-10866-z.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. DeepFM: A factorization-machine based neural network for CTR prediction. In *IJCAI*, pp. 1725–1731, 2017. doi: 10.24963/ijcai.2017/239.

Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Quoc Viet Hung Nguyen. Streaming session-based recommendation. In *KDD*, pp. 1569–1577, 2019. doi: 10.1145/3292500.3330839.

Xingzhuo Guo, Junwei Pan, Ximei Wang, Baixu Chen, Jie Jiang, and Mingsheng Long. On the embedding collapse when scaling up recommendation models. In *International Conference on Machine Learning (ICML)*, pp. 16891–16909. PMLR, 2024.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, pp. 544–560, 2020. doi: 10.1007/978-3-030-58517-4_32.

Bowei He, Xu He, Renrui Zhang, Yingxue Zhang, Ruiming Tang, and Chen Ma. Dynamic embedding size search with minimum regret for streaming recommender system. In *CIKM*, pp. 741–750, 2023. doi: 10.1145/3583780.3615135.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pp. 173–182, 2017. doi: 10.1145/3038912.3052569.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. LightGCN: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, pp. 639–648, 2020. doi: 10.1145/3397271.3401063.

Yicheng He, Zhou Kaiyu, Haoyue Bai, Fengbin Zhu, and Yonghui Yang. Understanding embedding scaling in collaborative filtering. *Transactions on Machine Learning Research (TMLR)*, 2025.

O. A. S. Ibrahim, E. M. G. Younis, E. A. Mohamed, et al. Revisiting recommender systems: An investigative survey. *Neural Computing and Applications*, 37:2145–2173, 2025. doi: 10.1007/s00521-024-10828-5.

Luo Ji, Qi Qin, Bingqing Han, and Hongxia Yang. Reinforcement learning to optimize lifetime value in cold-start recommendation. In *CIKM*, pp. 782–791, 2021. doi: 10.1145/3459637.3482292.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *ICDM*, pp. 197–206, 2018. doi: 10.1109/ICDM.2018.00035.

Julian Killingback, Mahta Rafiee, Madine Manas, and Hamed Zamani. Scaling laws for embedding dimension in information retrieval, 2026. URL `https://arxiv.org/pdf/2602.05062`.

Harold J Kushner and Dean S Clark. *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag, New York, 1978.

Yuanguo Lin, Yong Liu, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, and Chunyan Miao. A survey on reinforcement learning for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10):13164–13184, 2023.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search, 2019.

Haochen Liu, Xiangyu Zhao, Chong Wang, Xiaobing Liu, and Jiliang Tang. Automated embedding size search in deep recommender systems. In *SIGIR*, pp. 2307–2316, 2020. doi: 10.1145/3397271.3401436.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, volume 30, pp. 6467–6476, 2017.

Qinyi Luo, Penghan Wang, Wei Zhang, Fan Lai, Jiachen Mao, Xiaohan Wei, Jun Song, Wei-Yu Tsai, Shuai Yang, Yuxi Hu, and Xuehai Qian. FIITED: Fine-grained embedding dimension optimization during training for recommender systems, 2024.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, et al. Deep learning recommendation model for personalization and recommendation systems, 2019.

Yunke Qu, Liang Qu, Tong Chen, Xiangyu Zhao, Quoc Viet Hung Nguyen, and Hongzhi Yin. Scalable dynamic embedding size search for streaming recommendation. In *CIKM*, pp. 1941–1950, 2024. doi: 10.1145/3627673.3679638.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

N. Singh and S. K. Singh. A systematic literature review of solutions for cold start problem. *International Journal of System Assurance Engineering and Management*, 15:2818–2852, 2024. doi: 10.1007/s13198-024-02359-y.

Yifan Song, Peiyi Wang, Weimin Xiong, Dawei Zhu, Tianyu Liu, Zhifang Sui, and Sujian Li. InfoCL: Alleviating catastrophic forgetting in continual text classification from an information theoretic perspective. In *Findings of EMNLP*, pp. 14557–14570, 2023. doi: 10.18653/v1/2023.findings-emnlp.969.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pp. 1441–1450, 2019. doi: 10.1145/3357384.3357895.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

Jie Wang, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. Reinforcement learning-based recommender systems with large language models for state reward and action modeling. In *Proceedings of the 47th International ACM SIGIR conference on research and development in information retrieval*, pp. 375–385, 2024.

Jie Wang, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. Large language model driven policy exploration for recommender systems. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pp. 107–116, 2025.

Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. Streaming ranking based recommender systems. In *SIGIR*, pp. 525–534, 2018. doi: 10.1145/3209978.3210016.

Chaohua Yang, Dugang Liu, Xing Tang, Yuwen Fu, Xiuqiang He, Xiangyu Zhao, and Zhong Ming. Multi-scenario instance embedding learning for deep recommender systems. In *SIGIR*, pp. 2132–2141, 2025. doi: 10.1145/3726302.3730045.

Hyunsik Yoo, SeongKu Kang, Ruizhong Qiu, Charlie Xu, Fei Wang, and Hanghang Tong. Embracing plasticity: Balancing stability and plasticity in continual recommender systems. In *SIGIR*, pp. 2092–2101, 2025a.

Hyunsik Yoo, SeongKu Kang, and Hanghang Tong. Continual recommender systems. In *CIKM*, pp. 6857–6860, 2025b.

Zijian Zhang, Shuchang Liu, Ziru Liu, Rui Zhong, Qingpeng Cai, Xiangyu Zhao, Chunxu Zhang, Qidong Liu, and Peng Jiang. Llm-powered user simulator for recommender system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 13339–13347, 2025.

Xiangyu Zhao, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Xiwang Yang. AutoEmb: Automated embedding dimensionality search in streaming recommendations. In *ICDM*, pp. 896–905, 2021.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Jessie Li. Drn: A deep reinforcement learning framework for news recommendation. In *WWW*, 2018.

S. Zhou and Q. Li. Mitigating catastrophic forgetting in lifelong learning: A hybrid architecture integrating neural ordinary differential equations with memory-augmented transformers. *Scientific Reports*, 2026. doi: 10.1038/s41598-025-31685-9.

A THEORETICAL ANALYSIS OF TAMREC

This appendix provides supporting theoretical analysis for TAMRec, including policy convergence, unbiased and variance-reduced policy gradient, near-zero catastrophic forgetting, on-demand allocation optimality, sparse-feedback reward stabilization, bounded storage growth, and complexity analysis. All symbols are consistent with the main text.

A.1 STREAMING TASK FORMULATION

For the forgetting analysis, we formulate streaming recommendation as a sequence of chronological tasks:

$$\mathcal{T} = \{T_1, T_2, \ldots, T_M\},$$

where $\mathcal{T}$ denotes the set of sequential tasks constructed from the streaming interaction data, T_m denotes the m-th chronological task or time window, and M is the total number of sequential tasks.

Under the non-replayable streaming protocol, the model observes each batch once. For each incoming batch, the model first performs inference and records the evaluation results, and then updates its parameters using the corresponding ground-truth labels. This formulation is used only for continual-learning and forgetting analysis; the recommendation model itself is trained directly on the chronological interaction stream.

A.2 NOTATIONS

Table 6 lists all symbols used in the paper.

Table 6: Key Notations

Symbol	Description
	General Notations
M	Number of sequential tasks/time windows
t	Training time step
T	Total training steps
d_t	Embedding dimension at step t
$\bar{d}_T$	Average embedding dimension over T steps
$d_{\max}$	Maximum candidate embedding dimension
α	Upper bound coefficient ($\alpha < 1$) for average dimension growth
N_x, N_i	Interaction count of entity x / i
	Main Paper Notations
θ	Policy parameters
$\pi_\theta(a \mid s)$	Stochastic policy
$J(\theta)$	Expected long-term reward of policy π_θ
$\nabla_\theta J(\theta)$	True policy gradient
$\hat{\nabla}_\theta J(\theta)$	Estimated policy gradient
s_t	State at time t: $s_t = [f_t, q_t, \Delta_t]$
f_t	Cumulative interaction frequency up to t
q_t	Current embedding dimension index
Δ_t	Time-decayed interval feature
a_t	Action: $\{\text{Maintain}, \text{Enlarge}\}$
$r(s_t, a_t)$	Immediate reward signal
$b(s_t)$	Adaptive reward baseline
$\bar{L}_{\text{base},t}$	Baseline loss (general form of $b(s_t)$)
$b_{\text{group}}(s)$	Group-prior baseline for sparse entities
G_i	Peer group with same interaction count as entity i
L_t	Recommendation loss at step t
$L_{x,j}^{\text{hist}}$	Historical loss of entity x at step j
T_x	Number of historical loss records for entity x
$\Delta\tau_{x,j}^{\text{hist}}$	Time interval for historical loss
λ	Time-decay coefficient
β	Reward scaling coefficient
ϵ	Small constant for numerical stability
$\mathcal{D}$	Candidate embedding dimension set
$\hat{e}_u, \hat{e}_v$	Final user/item embeddings
$\Theta_{\text{RS}}, \Theta_{\text{Policy}}$	Parameters of recommendation/policy networks
	Appendix-Specific Notations
α_k	Learning rate at policy iteration k
g_t	Log-policy gradient term
σ_i^2, σ_G^2	Variance of loss estimates
η_t	Noise term in loss-accuracy relation
C	Constant offset in reward formulation
$U(d)$	Utility function of embedding dimension d
ΔU	Marginal utility gain of enlargement
d^*	Saturation dimension
d_i^∞	Asymptotic embedding dimension of entity i

A.3 THEOREMS AND PROOFS

Theorem 1 (Policy Convergence)**.** *Under Lipschitz continuous time-aware state transitions, uniformly bounded group-prior reward, and monotonic non-decreasing embedding updates, the policy sequence $\{\pi_k\}$ generated by TAMRec converges almost surely to a locally optimal deterministic policy π^*, i.e., $\pi_k \xrightarrow{a.s.} \pi^*$.*

Proof. Since q_t is non-decreasing and takes values in the finite index set $\{1, \ldots, max\}$, there exists $T_q < \infty$ such that $q_t = q^*$ for all $t \geq T_q$. The remaining state components $[f_t, \Delta_t]$ are bounded and evolve under a Lipschitz-continuous kernel, hence converge almost surely to a stationary distribution.

The policy update $\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta J(\theta_k)$ satisfies the Robbins–Monro conditions ($\sum \alpha_k = \infty$, $\sum \alpha_k^2 < \infty$)Kushner & Clark (1978). By the Kushner–Clark theorem Robbins & Monro (1951), $\nabla_\theta J(\theta_k) \xrightarrow{\text{a.s.}} 0$. Since $J(\theta)$ is smooth and the parameter space is compact, θ_k converges to a stationary point θ^*, which corresponds to a locally optimal policy $\pi^* = \pi_{\theta^*}$. Therefore $\pi_k \xrightarrow{\text{a.s.}} \pi^*$. □

Theorem 2 (Unbiased and Variance-Reduced Policy Gradient)**.** *Let $\pi_\theta(a_t \mid s_t)$ denote a parameterized stochastic policy, where $s_t \in \mathcal{S}$ is the state and $a_t \in \mathcal{A}$ is the action at step t. Let $r(s_t, a_t)$ denote the reward function, and let $b(s_t)$ denote an adaptive baseline that depends only on the state s_t. The policy gradient estimator*

$$\hat{\nabla}_\theta J(\theta) = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \big(r(s_t, a_t) - b(s_t)\big).$$

satisfies the following two properties:

1. ***Unbiasedness:*** *If $b(s_t)$ is independent of the sampled action a_t, then*
$$\mathbb{E}\left[\hat{\nabla}_\theta J(\theta)\right] = \nabla_\theta J(\theta).$$
2. ***Variance reduction:*** *The baseline reduces the gradient variance under sparse feedback. The optimal baseline minimizing the variance (in a weighted least-squares sense) is*
$$b^*(s) = \frac{\mathbb{E}\left[\|\nabla_\theta \log \pi_\theta(a_t \mid s_t)\|^2 \cdot r(s_t, a_t) \,\middle|\, s_t = s\right]}{\mathbb{E}\left[\|\nabla_\theta \log \pi_\theta(a_t \mid s_t)\|^2 \,\middle|\, s_t = s\right]}.$$

Proof. We prove the two properties separately.

Unbiasedness. By definition, the true policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot r(s_t, a_t)\right].$$

The estimator with baseline is

$$\hat{\nabla}_\theta J(\theta) = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \big(r(s_t, a_t) - b(s_t)\big).$$

Taking the expectation and splitting the sum, we obtain

$$\mathbb{E}\left[\hat{\nabla}_\theta J(\theta)\right] = \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot r(s_t, a_t)\right] - \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot b(s_t)\right].$$

The first term is exactly $\nabla_\theta J(\theta)$. For the second term, since $b(s_t)$ depends only on s_t and is independent of a_t, we apply the law of iterated expectation:

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot b(s_t)\right] = \mathbb{E}\left[b(s_t) \cdot \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \,\middle|\, s_t\right]\right].$$

By the log-derivative trick,

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \,\middle|\, s_t\right] = \int \nabla_\theta \log \pi_\theta(a \mid s_t)\, \pi_\theta(a \mid s_t)\, da = \nabla_\theta \int \pi_\theta(a \mid s_t)\, da = \nabla_\theta 1 = 0.$$

Thus the second term vanishes, and $\mathbb{E}\left[\hat{\nabla}_\theta J(\theta)\right] = \nabla_\theta J(\theta)$, i.e., the estimator is unbiased.

Variance reduction. Let $g_t = \nabla_\theta \log \pi_\theta(a_t \mid s_t)$ denote the log-policy gradient at step t. The gradient estimator is

$$\hat{\nabla}_\theta J(\theta) = \sum_t g_t \cdot \big(r(s_t, a_t) - b(s_t)\big).$$

The variance of the estimator is

$$\mathrm{Var}\left[\hat{\nabla}_\theta J(\theta)\right] = \mathbb{E}\left[\left\|\hat{\nabla}_\theta J(\theta)\right\|^2\right] - \left\|\mathbb{E}\left[\hat{\nabla}_\theta J(\theta)\right]\right\|^2.$$

By unbiasedness, $\mathbb{E}\left[\hat{\nabla}_\theta J(\theta)\right] = \nabla_\theta J(\theta)$, so

$$\mathrm{Var}\left[\hat{\nabla}_\theta J(\theta)\right] = \mathbb{E}\left[\left\|\sum_t g_t\big(r(s_t, a_t) - b(s_t)\big)\right\|^2\right] - \left\|\nabla_\theta J(\theta)\right\|^2.$$

For simplicity, we consider the per-sample contribution, where the variance depends on

$$\mathbb{E}\left[\|g_t\|^2 \left(r(s_t, a_t) - b(s_t)\right)^2\right].$$

The term $\|\nabla_\theta J(\theta)\|^2$ is independent of $b(s_t)$, so minimizing the variance reduces to minimizing the above expectation.

For each state s, define the objective

$$L(b) = \mathbb{E}\left[\|g_t\|^2 \left(r(s_t, a_t) - b\right)^2 \,\middle|\, s_t = s\right].$$

Differentiating with respect to b and setting the derivative to zero yields the optimal baseline

$$b^*(s) = \frac{\mathbb{E}\left[\|g_t\|^2 \cdot r(s_t, a_t) \,\middle|\, s_t = s\right]}{\mathbb{E}\left[\|g_t\|^2 \,\middle|\, s_t = s\right]}.$$

TAMRec's group-prior baseline $b(s_t) = \bar{L}_{\text{base},i}$ approximates this optimum by borrowing statistical strength from peer entities. Compared to the zero baseline ($b = 0$), it moves the baseline toward the conditional expectation of the reward, thereby reducing variance. □

Theorem 3 (Near-Zero Catastrophic Forgetting)**.** *Let M denote the total number of tasks, indexed by $j = 1, 2, \ldots, M$. Let $t \in \{1, \ldots, M\}$ denote the training time step, where task j is trained at step $t = j$. Let $V_j(t)$ denote the performance of the model on task j at time t, and let D_t denote the set of allocated embedding dimensions at time t.*

Under the monotonic non-decreasing embedding allocation strategy (Maintain and Enlarge), shrink-induced catastrophic forgetting is eliminated.

Proof. TAMRec only uses the Maintain and Enlarge operations, so the set of allocated embedding dimensions is non-decreasing over time:

$$D_{t+1} \supseteq D_t \quad \text{for all } t = 1, 2, \ldots, M-1.$$

For any historical task j and any time $t \geq j$, since $D_j \subseteq D_t$, all embedding dimensions required by task j remain allocated. Therefore, the performance on task j cannot decrease over time:

$$V_j(t) \geq V_j(j).$$

This implies $V_j(j) - V_j(M) \leq 0$, and consequently the positive part satisfies:

$$\big(V_j(j) - V_j(M)\big)_+ = \max\big(0, V_j(j) - V_j(M)\big) = 0.$$

Substituting into the definition of the Forgetting Measure (FM):

$$\text{FM} = \frac{1}{M-1} \sum_{j=1}^{M-1} \max_{k \geq j} \big(V_j(k) - V_j(M)\big)_+ = 0.$$

In shrink-induced methods, $D_{t+1} \subset D_t$ occurs for some t, causing $V_j(t+1) < V_j(t)$ for some historical task j, which yields FM > 0. Therefore, TAMRec achieves near-zero catastrophic forgetting. □

Theorem 4 (On-Demand Allocation Optimality)**.** *Under the accuracy–storage trade-off, TAMRec enlarges embedding dimension from d_k to d_{k+1} if and only if*

$$\mathrm{Acc}(d_{k+1}) - \mathrm{Acc}(d_k) \;>\; \delta \cdot c \cdot (d_{k+1} - d_k),$$

where $\mathrm{Acc}(d)$ is the prediction accuracy, c is the storage cost per dimension, and $\delta > 0$ is a trade-off coefficient.

Proof. Define the utility of dimension d as $U(d) = \mathrm{Acc}(d) - \delta \cdot c \cdot d$. The marginal utility of enlargement is

$$\Delta U = \big(\mathrm{Acc}(d_{k+1}) - \mathrm{Acc}(d_k)\big) - \delta \cdot c \cdot (d_{k+1} - d_k).$$

TAMRec's immediate reward is $r = \bar{L}_{\text{base},i} - L_{t,i}$. Assume the loss-accuracy relationship: $L_{t,i} = C - \mathrm{Acc}(d) + \eta_t$, where C is a constant and η_t has zero mean. Then

$$\mathbb{E}[r \mid \text{Enlarge}] = (\bar{L}_{\text{base},i} - C) + \mathrm{Acc}(d_{k+1}),$$

$$\mathbb{E}[r \mid \text{Maintain}] = (\bar{L}_{\text{base},i} - C) + \mathrm{Acc}(d_k).$$

Taking the difference:

$$\mathbb{E}[r \mid \text{Enlarge}] - \mathbb{E}[r \mid \text{Maintain}] = \mathrm{Acc}(d_{k+1}) - \mathrm{Acc}(d_k).$$

Thus $\mathbb{E}[r \mid \text{Enlarge}] > \mathbb{E}[r \mid \text{Maintain}]$ if and only if $\Delta U > 0$. The policy selects Enlarge exactly when this condition holds, achieving on-demand optimality. □

Theorem 5 (Sparse-Feedback Reward Stabilization)**.** *For sparse-feedback entities with $N_i < K$, the group-prior baseline*

$$b_{group}(s) = \frac{1}{|G_i|} \sum_{m \in G_i} \bar{L}^{(m)}, \quad G_i = \{m \mid N_m = N_i\}.$$

provides a stable reward signal and mitigates cold-start reward instability.

Proof. When $N_i < K$, entity i has insufficient historical loss records to compute a reliable baseline from its own history. TAMRec uses the same-frequency peer group G_i to compute a fallback reference.

The variance of the individual estimate is $\mathrm{Var}(\bar{L}_i) = \sigma_i^2 / T_i$, where T_i is the number of historical samples. The variance of the group prior is $\mathrm{Var}(b_{\text{group}}) = \sigma_G^2 / |G_i|$, where σ_G^2 is the variance among peer entities.

In sparse-feedback regimes, T_i is small (typically 1 to 5) while $|G_i|$ can be significantly larger. Since $|G_i| \gg T_i$ in most recommendation scenarios, we have $\mathrm{Var}(b_{\text{group}}) \ll \mathrm{Var}(\bar{L}_i)$. This reduces reward variance during cold start. Moreover, by Theorem 2, the group-prior baseline does not introduce bias in the policy gradient because it remains independent of the sampled action.

Please note that $b(s_t) = \bar{L}_{\text{base},t}$ denotes the general form of our adaptive reward baseline, and $b_{\text{group}}(s)$ serves as its concrete implementation under the sparse feedback scenario. □

Theorem 6 (Bounded Storage Growth)**.** *Under the performance-improvement reward and time-aware policy, the average embedding dimension satisfies*

$$\limsup_{T \to \infty} \bar{d}_T \;\le\; \alpha \cdot d_{\max}, \qquad \alpha < 1,$$

where $\bar{d}_T = \frac{1}{T} \sum_{t=1}^{T} d_t$ and $d_{\max}$ is the maximum candidate dimension.

Proof. By Theorem 4, the Enlarge action is selected only when $\Delta\mathrm{Acc} > \delta \cdot c \cdot \Delta d$, where $\Delta\mathrm{Acc} = \mathrm{Acc}(d_{k+1}) - \mathrm{Acc}(d_k)$ and $\Delta d = d_{k+1} - d_k$.

As the embedding dimension increases, the marginal accuracy gain $\Delta\mathrm{Acc}$ decays to zero due to representational saturation. Therefore, there exists a saturation point $d^* < d_{\max}$ such that for all $d \ge d^*$, the inequality $\Delta\mathrm{Acc} > \delta \cdot c \cdot \Delta d$ fails.

Additionally, the time-decayed interval feature Δ_t further suppresses unnecessary expansion by distinguishing temporally redundant frequent interactions from genuinely informative ones. Consequently, for each entity, the embedding dimension converges almost surely to some limiting value $d_i^{\infty} \leq d^*$.

The asymptotic average dimension satisfies

$$\limsup_{T\to\infty} \bar{d}_T \;=\; \limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} d_t \;\leq\; d^*.$$

Define $\alpha = d^*/d_{\max}$. Since $d^* < d_{\max}$, we have $\alpha < 1$. Therefore

$$\limsup_{T\to\infty} \bar{d}_T \;\leq\; \alpha \cdot d_{\max}, \quad \alpha < 1.$$

□

A.4 Complexity Analysis

Space Complexity.

Fixed-Dimension:

$$\mathcal{O}(N \cdot d_{\max} + |\Theta_{RS}|).$$

TAMRec:

$$\mathcal{O}(N \cdot \bar{d} + |\Theta_{RS}| + |\Theta_{\text{Policy}}|).$$

Because $|\Theta_{\text{Policy}}|$ is independent of N and typically negligible:

$$\mathcal{O}(N \cdot \bar{d}) \ll \mathcal{O}(N \cdot d_{\max}).$$

Time Complexity (Per Batch).

Fixed-Dimension:

$$\mathcal{O}(|B| \cdot L \cdot d_{\max}^2).$$

TAMRec:

$$\mathcal{O}(|B| \cdot C_{\text{policy}}) + \mathcal{O}(|B| \cdot L \cdot \bar{d}^2),$$

where C_{policy} denotes the constant per-sample computational cost introduced by the lightweight policy network, L is the number of layers in the recommendation backbone, and $\bar{d}$ is the average allocated embedding dimension.

Since the policy network is lightweight and independent of the total number of users and items, C_{policy} is much smaller than the main recommendation computation. Therefore, TAMRec maintains near-constant per-batch runtime while reducing embedding storage from $\mathcal{O}(Nd_{\max})$ to $\mathcal{O}(N\bar{d})$.

B Implementation Details

This appendix provides the detailed implementation settings of TAMRec, including hyperparameters, training protocol, and evaluation settings.

B.1 Model Configuration

Embedding Search Space. We define a discrete candidate embedding dimension set for both user and item representations:

$$\mathcal{D} = \{4, 8, 16, 32, 64, 128\}.$$

Recommendation Backbone. We adopt a standard Multi-Layer Perceptron (MLP) as the backbone recommender. The backbone is optimized using Adam with an initial learning rate of 10^{-4}.

Policy Network. The window size K, the time-decay coefficient λ, and the reward-decay coefficient β are treated as hyperparameters and selected through preliminary tuning on the validation stream. In our experiments, we use $K = 5$, $\lambda = 5$, and $\beta = 0.5$ as the default configuration across datasets. The numerical stability constant is fixed as $\epsilon = 10^{-8}$. The policy network is optimized using Adam with learning rate 10^{-4}.

Training Setup. The overall framework is trained with mini-batch size 500.

B.2 Warm-Up Strategy

To reduce reward instability caused by random initialization in early training, we employ a warm-up strategy inspired by prior NAS and RL-based recommendation works Guo et al. (2020); Zheng et al. (2018). During the first $W = 500$ mini-batches, the policy network remains frozen while the recommender backbone updates normally. This allows TAMRec to accumulate stable state statistics and group priors before activating dynamic dimension allocation.

B.3 Streaming Evaluation Protocol

Unlike traditional offline recommendation, streaming recommendation does not strictly separate training and testing. Although all datasets are chronologically partitioned into train/test splits, evaluation is performed in an online continual-learning manner.

Specifically, for each incoming batch in the test stream:

1. The current model first generates predictions;
2. Evaluation metrics are computed;
3. The model is immediately updated using the newly observed interactions.

All baseline methods follow the identical streaming protocol to ensure fair comparison.

B.4 Evaluation Metrics

Explicit Feedback Datasets. For MovieLens and Netflix datasets, we evaluate predictive performance using:

- Accuracy
- Cross-Entropy (CE) Loss

Implicit Feedback Streaming Datasets. For Amazon-Beauty and LastFM-1K, we formulate recommendation as a Top-K ranking task and report:

- HR@10, HR@20
- NDCG@10, NDCG@20

Table 7: Chronological Train/Test Split of Explicit Feedback Datasets

Dataset	Train Set	Test Set
Movielens-20m	15,000,000	5,000,000
Movielens-latest	26,000,000	7,000,000
Netflix Prize	80,000,000	20,000,000
Amazon-beauty	225,788	56,448
Lastfm-1k	10,291,057	2,572,765

C RQ7: COMPUTATIONAL EFFICIENCY ANALYSIS

To quantify the practical overhead introduced by TAMRec's policy network, we compare TAMRec against multiple fixed-dimension baselines on the Amazon Beauty dataset under identical hardware and optimization settings.

C.1 EXPERIMENTAL SETUP

All experiments are conducted on a workstation with an NVIDIA RTX 5060 GPU with 8GB memory, an AMD Ryzen 5 9600X CPU, and 32GB RAM. All models are trained using:

- Batch size: 1,024
- Learning rate: 0.001
- Training epochs: 1

We evaluate four configurations:

1. **TAMRec**: Full model with dynamic policy network
2. **Fixed-128**: Static embeddings with dimension 128
3. **Fixed-32**: Static embeddings with dimension 32
4. **Fixed-4**: Static embeddings with dimension 4

For fixed baselines, the policy network is disabled.

C.2 OVERALL RUNTIME COMPARISON

Table 8 reports the total runtime, training time, and testing time.

TAMRec achieves nearly identical total runtime to Fixed-128 (16.61 min vs. 16.85 min, -1.47%), while using substantially smaller average embedding sizes (User: 4.5, Item: 16.3). This corresponds to approximately:

- 96.5% user embedding reduction
- 87.3% item embedding reduction

Although policy control introduces extra training overhead, TAMRec's smaller embeddings accelerate downstream recommendation computation, offsetting the controller cost.

Table 8: Computational efficiency comparison on Amazon Beauty.

Method	User_{dim}	Item_{dim}	T_{total}(min)	T_{train}(min)	T_{test}(min)
TAMRec	4.5	16.3	16.61	0.41	16.19
Fixed-128	128.0	128.0	16.85	0.36	16.49
Fixed-32	32.0	32.0	16.90	0.36	16.54
Fixed-4	4.0	4.0	16.85	0.37	16.48

C.3 PER-BATCH OVERHEAD BREAKDOWN

To further analyze overhead sources, Table 9 presents the detailed per-batch runtime decomposition.

Compared with Fixed-128:

- Total extra overhead: +13.06 ms/batch
- Controller update (REINFORCE + reward): +11.62 ms
- RS-side policy lookup and embedding transfer: +1.45 ms

Importantly, the RS backward pass remains nearly unchanged:

$$18.26 \text{ ms (TAMRec)} \approx 18.60 \text{ ms (Fixed-128)}$$

This demonstrates that TAMRec's policy mechanism mainly affects lightweight control modules without significantly slowing the main recommender optimization.

Table 9: Per-batch time breakdown of TAMRec (ms).

Component	Description	Time
RS Forward	Recommendation forward pass	2.04
RS Backward	Recommendation backward pass	18.26
Policy Fwd(Ctrl)	Policy inference for dimension control	2.12
Policy Fwd(RS)	Policy lookup during RS update	1.73
Emb Transfer(Ctrl)	Dimension-level embedding transfer	1.40
Emb Transfer(RS)	RS-side embedding alignment	1.54
Controller Total	REINFORCE reward + policy gradient	11.62

C.4 Conclusion

Overall, TAMRec introduces only a modest control overhead while preserving near-constant total runtime. This validates that dynamic embedding allocation can be deployed in streaming systems without sacrificing practical efficiency.

D RQ8: HYPERPARAMETER SENSITIVITY ANALYSIS

D.1 EXPERIMENTAL SETUP

We vary one hyperparameter at a time on Amazon Beauty, using the same streaming protocol as the main experiments. When varying K we fix $\lambda = 5$ and $\beta = 0.5$; when varying λ we fix $K = 5$ and $\beta = 0.5$; when varying β we fix $K = 5$ and $\lambda = 5$. The candidate set is $\{4, 8, 16, 32, 64, 128\}$ throughout. The default configuration is shown in bold in Tables 10–12.

Table 10: Effect of reward window size K on Amazon Beauty.

K	HR@10 ↑	NDCG@10 ↑	HR@20 ↑	NDCG@20 ↑	Item Ratio ↓
1	0.1291	0.0698	0.1642	0.0786	0.1744
3	0.1311	0.0714	0.1679	0.0807	0.1388
5	**0.1316**	**0.0713**	**0.1700**	**0.0810**	**0.1219**
10	0.1308	0.0687	0.1691	0.0784	**0.0890**

Table 11: Effect of time-decay coefficient λ on Amazon Beauty.

λ	HR@10 ↑	NDCG@10 ↑	HR@20 ↑	NDCG@20 ↑	Item Ratio ↓
1	0.1316	0.0716	0.1704	0.0814	0.1249
2	0.1322	0.0714	0.1704	0.0810	0.1272
5	**0.1316**	**0.0713**	**0.1700**	**0.0810**	**0.1219**
10	0.1312	0.0713	0.1698	0.0811	0.1295
20	0.1307	0.0710	0.1694	0.0808	0.1291

Table 12: Effect of reward-decay coefficient β on Amazon Beauty.

β	HR@10 ↑	NDCG@10 ↑	HR@20 ↑	NDCG@20 ↑	Item Ratio ↓
0	0.1316	0.0713	0.1700	0.0810	0.1219
0.3	0.1314	0.0715	0.1699	0.0812	0.1228
0.5	**0.1316**	**0.0713**	**0.1700**	**0.0810**	**0.1219**
0.7	0.1312	0.0711	0.1696	0.0809	0.1233
1.0	0.1305	0.0706	0.1689	0.0803	0.1252

D.2 ACCURACY

Across all settings, HR@10 spans only 0.1291–0.1322 and NDCG@10 only 0.0687–0.0716. The largest gap from the default is 0.0024 for K (at $K = 1$), 0.0006 for λ (at $\lambda = 20$), and 0.0011 for β (at $\beta = 1.0$). All three gaps are therefore at or below 0.0024 in absolute HR@10. None of the three sweeps shows a clear trend, so we do not read small gaps (such as the slightly higher HR@10 at $\lambda = 2$) as a better setting.

D.3 EFFECT OF WINDOW SIZE

Accuracy stays flat, but K controls storage. The item ratio falls steadily:

$$0.1744\ (K{=}1)\ \rightarrow\ 0.1388\ \rightarrow\ 0.1219\ \rightarrow\ 0.0890\ (K{=}10),$$

a 49.0% cut, far larger than any accuracy fluctuation observed across the sweep. The reason follows from the reward design: a larger K averages the baseline over more past losses, so advantage estimates are smoother and *Enlarge* fires less often.

This gives a practical knob. At $K = 10$, item embeddings take 27.0% less space than the default while HR@10 differs by only 0.0008. Under a tight memory budget, raising K costs almost nothing in accuracy.

D.4 Effect of Decay Coefficients

For λ, the item ratio stays between 0.1219 and 0.1295 with no trend. Attention appears to already rescale Δ_t, so the policy is robust to the exact value.

β behaves similarly: the item ratio stays within 0.1219–0.1252 and accuracy within 0.1305–0.1316. Weighting recent losses more or less heavily has little effect once the window average is already stable, which is consistent with K, not the weighting scheme, being the factor that shapes allocation.

D.5 Conclusion

TAMRec's accuracy is robust to all three hyperparameters and needs no per-dataset tuning. K also gives a predictable way to trade embedding space for a negligible accuracy loss, which is useful when memory is limited.

E RQ6: EVALUATION OF CATASTROPHIC FORGETTING

To rigorously quantify the impact of embedding shrink operations on historical information retention, we adopt a standard continual learning (CL) evaluation protocol following prior works Chaudhry et al. (2018); Lopez-Paz & Ranzato (2017).

E.1 EXPERIMENTAL SETUP

We model streaming recommendation as a sequence of chronological tasks:

$$\mathcal{T} = \{T_1, T_2, \ldots, T_M\},$$

where each task corresponds to one temporal segment of the interaction stream.

Let $A_{k,j}$ denote the performance on task T_j after training through task T_k ($k > j$).

Backward Transfer (BWT). BWT measures how future learning influences historical tasks:

$$BWT = \frac{1}{M-1} \sum_{j=1}^{M-1} (A_{M,j} - A_{j,j}).$$

Negative BWT indicates catastrophic forgetting, while positive BWT indicates beneficial knowledge transfer.

Forgetting Measure (FM). FM captures the maximum historical performance degradation:

$$FM = \frac{1}{M-1} \sum_{j=1}^{M-1} \max_{l \in \{1,\ldots,M-1\}} (A_{l,j} - A_{M,j}).$$

Lower FM indicates better memory retention.

We partition the streaming data into five chronological tasks ($T_1 \sim T_5$) and evaluate HR@10 and NDCG@10.

E.2 RESULTS

Table 5 in the main text reports the complete results.

E.3 ANALYSIS

Impact of Shrink Operations. SCALL, the shrink-based method, produces negative BWT and significantly larger FM. This demonstrates that forced structural parameter removal causes irreversible forgetting in streaming settings, especially without replay mechanisms.

Benefits of Monotonic Capacity. All monotonic-capacity methods (Fixed-128, ESAPN, TAMRec) achieve near-zero FM, confirming that preserving allocated capacity fundamentally protects historical knowledge.

Positive BWT in recommender systems. Interestingly, shrink-free methods all show positive BWT. This differs from many traditional CL settings and likely arises from collaborative filtering effects: newly observed user-item interactions improve shared representations, indirectly benefiting older tasks.

Why TAMRec Performs Best. Although Fixed-128 avoids forgetting, its lower BWT suggests over-parameterization on sparse early-stage data may reduce future transferability. ESAPN improves transfer through dynamic growth but still tends toward frequency-driven parameter inflation.

TAMRec achieves the best balance:

- Highest final accuracy
- Highest BWT

- Zero forgetting

Its Time-Aware On-Demand design selectively expands capacity only when necessary, preserving compactness while maximizing transferable collaborative knowledge.

E.4 CONCLUSION

These findings strongly validate our central claim: shrink operations are fundamentally harmful in streaming recommendation, while monotonic time-aware expansion provides both anti-forgetting guarantees and stronger long-term generalization.

F RQ5: COLD-START AND LONG-TAIL ANALYSIS

To evaluate TAMRec's robustness under sparse-feedback conditions, we analyze recommendation performance across users with different activity levels, ranging from cold-start users to highly active users.

Figure 3 in the main text shows the activity-group comparison.

Figure 3 reports recommendation accuracy under varying user interaction frequencies.

Cold-Start Users (1–5 interactions). TAMRec achieves the strongest performance (0.3421), outperforming:

- SCALL: 0.3352
- ESAPN: 0.3287

This indicates that TAMRec's Group Prior reward stabilizes policy learning when explicit feedback is extremely limited.

Long-Tail Users (6–20 interactions). TAMRec maintains a consistent advantage (0.3988), while shrink-based methods begin to lose useful sparse historical signals due to forced parameter truncation.

Active Users. As interaction history grows, all methods improve and the performance gap narrows. Nevertheless, TAMRec remains competitive, showing that its robustness in sparse regimes does not sacrifice performance for dense users.

F.1 DISCUSSION

These findings validate the design motivation of TAMRec:

- **Group Prior Reward**: Reduces reward variance in sparse settings
- **Time-Aware Design**: Distinguishes meaningful sparse behaviors from noise
- **shrink-free Policy**: Preserves rare but valuable long-tail signals

Overall, TAMRec is particularly effective for cold-start and long-tail populations, where stable early-stage allocation is most critical.

G LIMITATIONS AND BROADER IMPACT

G.1 LIMITATIONS

Although TAMRec has demonstrated clear effectiveness in mitigating catastrophic forgetting and achieving constrained, on-demand embedding dimension growth, it still has several limitations that require improvement:

Unverified Applicability in Large-Scale Recommendation Scenarios: Currently, the maximum dimension d_{max} of the model is preset, and the experimental testing range only reaches up to 128 dimensions. While this dimension range aligns with common practices in the field of recommendation systems, actual industrial recommendation systems often demand dimensions exceeding this value. Due to limitations in laboratory resources and time constraints, we have not conducted tests on extremely high-dimensional scenarios. Therefore, significant work remains to be done if we aim to apply TAMRec's design philosophy to industrial-scale streaming recommendation systems.

Fixed Expansion Step Size Restricts Allocation Effectiveness: In terms of dimension expansion strategy, TAMRec only supports discrete step sizes for dimension enlargement. This setup may result in suboptimal allocation of embedding dimensions for entities whose optimal dimensions fall precisely between preset levels.

Insufficient Research on Cross-Domain Generalization Capability: At present, the TAMRec framework has only been validated on collaborative filtering datasets. Nevertheless, the core idea presented in the paper "Time-Aware and Monotonic Reinforcement Learning for On-Demand Embedding Dimensions" holds potential for application in a broader range of industrial scenarios that employ deep learning techniques. However, as of now, in-depth and systematic research on the framework's generalization capability in cross-domain scenarios has not been conducted.

G.2 SOCIETAL IMPACT

The core idea presented in this paper holds potential for application in a broader range of industrial scenarios that employ deep learning techniques. The monotonic, time-aware, and on-demand resource allocation paradigm is not restricted to recommendation tasks; it can be generalized to other deep learning-based dynamic systems such as online advertising, sequential modeling, lifelong learning, and large-scale retrieval systems. By providing an effective strategy to balance model expressiveness, efficiency, and stability, this work offers a practical and sustainable direction for building scalable, robust, and cost-effective industrial AI systems.

TAMRec's on-demand embedding allocation brings meaningful societal and environmental benefits by improving the efficiency and sustainability of real-time recommendation systems. By dynamically assigning embedding dimensions based on actual demand and avoiding unnecessary expansion, the method reduces storage consumption, memory usage, and computational overhead. This efficiency lowers energy consumption and carbon footprints in large-scale data centers, while making recommendation services more accessible on resource-constrained devices such as mobile and edge platforms. Furthermore, the framework's ability to handle long-tailed and cold-start entities helps promote fair and inclusive recommendation, reducing bias toward popular items and improving user experience for diverse groups.

H Full Ablation Results

These tables retain the complete component ablations for the explicit- and implicit-feedback datasets.

Table 13: Ablation study on explicit-feedback datasets. U/I Ratio denotes user/item embedding ratio; Acc./Loss reports accuracy and cross-entropy loss.

Method	MovieLens-20M		MovieLens-Latest		Netflix	
	U/I Ratio ↓	Acc./Loss	U/I Ratio ↓	Acc./Loss	U/I Ratio ↓	Acc./Loss
TAMRec	0.1401/0.1683	**0.4540/1.2486**	**0.0892**/0.2185	**0.4445**/1.2721	0.5579/**0.6251**	**0.4859/1.1551**
w/o Time-Aware	0.2286/**0.1676**	0.4537/1.2515	0.1303/0.3760	0.4413/**1.2716**	**0.4270**/0.9997	0.4201/1.2885
w/o Attention.	**0.0588**/0.6373	0.4517/1.2539	0.1877/0.1496	0.4435/1.2678	0.5063/0.7944	0.4833/1.1612
w/o Group Prior	0.2936/0.2936	0.4526/1.2572	0.2388/**0.1100**	0.4420/1.2695	0.7807/0.9973	0.4813/1.1642

Table 14: Ablation study on implicit-feedback streaming datasets. User/Item Ratio denotes the average allocated embedding ratio.

Amazon-Beauty						
Method	HR@10 ↑	NDCG@10 ↑	HR@20 ↑	NDCG@20 ↑	User Ratio ↓	Item Ratio ↓
TAMRec	**0.1316**	**0.0713**	**0.1700**	**0.0810**	0.0365	**0.1219**
w/o Time-Aware	0.1043	0.0576	0.1471	0.0684	0.0363	0.1729
w/o Attention	0.0962	0.0521	0.1427	0.0638	0.0362	0.1756
w/o Group Prior	0.1042	0.0568	0.1501	0.0682	0.0527	0.2015

LastFM-1K						
Method	HR@10 ↑	NDCG@10 ↑	HR@20 ↑	NDCG@20 ↑	User Ratio ↓	Item Ratio ↓
TAMRec	**0.1892**	**0.1101**	**0.2621**	**0.1285**	**0.9900**	**0.5548**
w/o Time-Aware	0.1569	0.0913	0.2174	0.1066	0.9956	0.7363
w/o Attention	0.1595	0.0929	0.2198	0.1081	0.9956	0.7381
w/o Group Prior	0.1623	0.0947	0.2286	0.1109	0.9978	0.7524