# Drift-Aware Multi-Pass Streaming Training for Generative Recommendation on Non-Stationary User Sequences

Aowei Shen[1], Minjae Rhee[2], Changhao Zhang[3,*], Yinuo Wang[3]

[1] University of Washington, Seattle, USA

[2] University of Illinois Urbana-Champaign, Champaign, USA

[3] University of Chicago, Chicago, USA

[*]Corresponding email: changhao.zhang@ieee.org

## Abstract

**Generative recommenders that represent an item as a short tuple of semantic tokens and decode the next item autoregressively have become a competitive alternative to embedding-based retrieval, but they are almost always studied on a static snapshot of the data. Deployed streams are not static: catalogues rotate, popularity bursts and user intent shifts, so the recommender must be re-trained repeatedly as blocks of interactions arrive. This paper asks how that re-training should be scheduled. We first run a block-level probe study on two real streams, forking the model at every update cycle and measuring how next-block accuracy responds to the number of optimisation passes and to the fraction of replayed history. The two streams disagree: on a fast-rotating stream (21% of the items in an average block are new) replaying half of the batch from memory costs 28% Recall@10 and three passes are enough, whereas on a slow stream replay helps and five passes still improve accuracy. No fixed schedule is therefore correct across streams, and per-stream tuning is impossible online. We propose DAMP, which estimates drift online from the divergence of the incoming block combined with the model's excess loss on that block, and uses a single drift score to gate four coupled decisions: the number of passes, the replay ratio, the distillation strength, and which memory samples are still relevant. On two real streams, DAMP is statistically tied with the strongest fixed schedule on the volatile one ($p=0.772$) while beating every stability-oriented baseline there, and on the stable one it is significantly better than all eight baselines (18.6% over fixed multi-pass training, $p<0.001$), including at matched compute. No fixed schedule is strong on both. We also report where the controller fails: on a synthetic generator whose item-frequency divergence is large while its preference structure stays partly stationary, our absolute drift calibration closes the replay gate when it should not. Code, data pipeline and raw logs are released.**

## Keywords

**Generative recommendation, semantic IDs, streaming training, concept drift, continual learning, recommender systems.**

## 1. Introduction

Sequential recommenders predict the next item from a user's interaction history, and for a decade the dominant designs have been recurrent [1] or self-attentive [2], [3] encoders followed by a dot product against a full item-embedding table. A newer family replaces that table with generation: each item is assigned a short tuple of discrete semantic tokens, and the model autoregressively decodes the tokens of the next item, with a prefix trie restricting decoding to tuples that correspond to real items [4]. The idea was imported from differentiable

search indices [5] and language-model-style recommendation [6], and has since been scaled to industrial systems [7] and extended with collaborative [8] and ranking-oriented [9] tokenisers. Because tokens are shared across items, the parameter count no longer grows linearly with the catalogue, and an item that has never been interacted with is still representable as soon as its metadata exists.

Almost all of this work evaluates a model trained once on a static snapshot. Production streams are not static. Catalogue turnover, release cycles, seasonality and shifting intent all make the joint distribution of (history, next item) drift, which is precisely the setting that industrial generative recommenders identify as their operating regime [7]. Our own measurements make the magnitude concrete: on the MovieTweetings stream [10] restricted to 2016--2018, only 32% of the items are present in the warm-up block, 21% of the items appearing in an average block have never been seen before, and only 52.7% of the ground-truth targets are even reachable from the catalogue known at serving time.

Substantial effort has gone into making the deployment of such models efficient on the serving side. CAGR is a good illustration of how far that can be pushed: by learning a hardware-aware graph-rewriting policy that transfers across NVIDIA, AMD and TPU backends, it recovers most of hand-tuned latency for DLRM-style recommenders in seconds instead of hours of per-platform auto-tuning [11]. Our question is the complementary one. Given that the model itself has to be re-fitted as the stream moves, how should that re-fitting be scheduled?

When a recommender is updated block by block, two decisions dominate both cost and quality: how many optimisation passes to spend on the incoming block, and how much of the past to mix back in. The streaming-recommendation literature answers the second question with tools inherited from continual learning - exemplar replay, knowledge distillation and parameter anchoring [12], [13], [14], [15], [16] - whose stated purpose is to reduce forgetting. That purpose presupposes that the past is still valid. Under strong drift a large part of the past is obsolete, and forgetting it is not a failure mode but the objective. The first question is usually not asked at all: the number of passes per update cycle is a hand-set constant, even though warm-started re-optimisation is known to interact badly with generalisation [17].

This paper studies both questions empirically and turns the answer into an online controller. Our contributions are:

1) A block-level probe study for a semantic-ID generative recommender. At every update cycle we fork the model and re-train it with 1--6 passes and with replay ratios 0--0.5, then measure next-block accuracy. The optimum flips between streams: on the fast-rotating stream replay is harmful and three passes suffice; on the slow stream replay helps and more passes keep helping. This directly contradicts the implicit assumption that a stability-oriented schedule is universally safe.

2) DAMP (Drift-Aware Multi-Pass), a controller that computes one online drift score per block from a distributional divergence plus the model's excess loss, and uses it to gate four coupled decisions: pass count, replay ratio, distillation weight, and relevance-based re-sampling of memory. The score costs two short forward passes and one histogram per block.

3) A prequential evaluation on two real and one synthetic stream with three seeds, eight baselines, explicit compute accounting, and ablations, plus a transfer check on an item-ID backbone. DAMP is best or tied-best on every stream while each fixed baseline wins on one stream and loses on another.

## 2. Related Work

### 2.1. Generative Recommendation with Semantic IDs

Vector quantisation [18] and its residual variant [19] turn a continuous item embedding into a short tuple of codes. TIGER [4] applied this to recommendation: items are tokenised from content embeddings with an RQ-VAE, and a sequence-to-sequence model generates the next item's tuple under trie-constrained beam search, which improves cold-start behaviour because a new item inherits tokens from its content. Related lines learn quantised item representations for transfer [20], [21], compress large catalogues by joint product quantisation [22], inject collaborative signal into the tokeniser [8], or study semantic IDs for ranking [9]. Residual k-means is a common lightweight substitute for the RQ-VAE tokeniser and is what we use. None of these works studies how the generative model should be re-trained as the stream evolves, which is our subject.

### 2.2. Streaming and Incremental Recommendation

Classical streaming recommenders update matrix factorisation incrementally [23] or maintain a reservoir of informative interactions [24], [25]. For neural session-based models, streaming session-based recommendation [26] and ADER [12] update the model in periodic cycles; ADER combines an exemplar set with a distillation loss whose weight grows with the cycle index, and reports that it can even beat re-training on all history. Memory-augmented incremental models [13] follow the same stability-oriented philosophy. The underlying toolbox comes from continual learning: parameter anchoring [14], function anchoring [15], [27], exemplar replay [16], [28], gradient projection [29] and logit replay [30]. All of these methods take the strength of the stability term as a fixed hyper-parameter. We keep the same primitives but make their strength a function of measured drift, and we add the pass budget as a controlled variable.

The assumption that replayed history is still valid has itself come under scrutiny, and the graph-learning literature offers the sharpest version of the argument. GT-ACGL studies continual learning when drift is adversarial rather than stochastic and makes the valuable observation that random replay sampling is statistically inefficient on imbalanced streams: by selecting buffer entries according to motif participation and stability under perturbation instead of by centrality or chance, it holds the observed forgetting rate below six percentage points at a modest training overhead [31]. Its lesson - that which history is retained matters at least as much as how much of it - directly motivates the relevance-weighted memory of Sec. IV-C.

A parallel trend treats memory as a first-class learnable object rather than as fixed infrastructure. LS-CM makes this unusually explicit for software-engineering agents, separating a per-episode retrieval buffer from a compressed parametric memory of strategies and letting a lightweight, gradient-free controller pick configurations from execution feedback [32]. We borrow the design discipline rather than the mechanism: our controller is likewise cheap, decoupled from the backbone it steers, and driven by a scalar measured online.

### 2.3. Concept Drift

Learning under drift is a mature field [33], [34], with detectors such as adaptive windowing [35] typically used as triggers: when a test fires, the model is retrained or the window is reset. Our use of drift is different in two ways. First, the signal is continuous rather than a binary alarm, because in recommendation drift is chronic rather than episodic. Second, it does not decide whether to retrain - in a streaming recommender one always retrains - but modulates how much plasticity to buy on this cycle.

Agents acting on the live web face the same non-stationarity in an acute form, and TAPE gives a treatment that is close to ours in spirit. It maintains a dual-stream experience pool and curates it with novelty-weighted retention, quality reweighting and an explicit staleness decay, and

reports the instructive result that performance is limited by the diversity of the retained experience rather than by its volume [36]. The principle that old experience should be down-weighted as a function of its age, not merely capped in quantity, is the closest analogue we know of to the drift-gating we apply to replay.

## 3. Problem Setting

Interactions arrive as a time-ordered stream and are partitioned into 10 blocks of equal volume by timestamp quantiles. Block t is split chronologically into an update slice D_t (the earlier 65%) and a serving slice Q_t (the latest 35%); serving slices are never trained on. We follow the prequential, test-then-train protocol used by streaming recommenders [12]: the model theta_{t-1}, which has seen D_1..D_{t-1}, is evaluated on Q_t, and only afterwards is it updated on D_t. Reported accuracy is the mean over blocks t = 2..T, so every reported number is an out-of-sample prediction of a future period.

A training example is a (history, target) pair built from a user's chronologically preceding items, truncated to the last 16. Constrained decoding is restricted to the catalogue known at serving time, i.e. items observed in D_1..D_{t-1}; targets outside it are counted as misses, which is why catalogue coverage in Table I upper-bounds recall. We report Recall@K and NDCG@K for K in {10, 20}. Cost is measured as the number of gradient updates, and a retrospective score (accuracy of the final model on all past serving slices) measures how much the model has forgotten.

## 4. METHOD

### 4.1. Generative Backbone

Each item is described by a short metadata string (title tokens, genres, release decade). We fit a TF-IDF vectoriser and a truncated SVD on the warm-up catalogue only, project every item to a unit vector, and quantise it with residual k-means: three levels of 32 centroids plus a fourth de-duplication level of at most 48 slots, giving each item a tuple (c1, c2, c3, c4). Because the codebooks are frozen after the warm-up block, an item that appears later in the stream is tokenised without re-fitting, which is the property that makes semantic IDs attractive under catalogue turnover [4].

An item's input embedding is the sum of its four codebook vectors, so new items are representable with no new parameters. A two-layer causal Transformer [37] encodes the history and its last state h conditions a GRU decoder that emits the four codes in order; training minimises the mean cross-entropy over the four levels. At inference we run beam search (width 24) over the code trellis with a prefix trie built from the known catalogue, so every beam ends on a real item. This is a deliberately small model (d = 48, two layers, two heads, history length 16, about 0.1 M parameters) because our object of study is the training schedule, not the backbone; all compared methods share it exactly, and the whole study runs on a single CPU core.

### 4.2. Online Drift Estimation

Before training on D_t we compute one scalar. Let p_t be the item-frequency distribution of D_t and p~_{t-1} an exponential moving average of previous blocks; let l_t^new and l_t^mem be the current model's mean loss on a sample of D_t and on a sample of the replay memory. Then

$$\delta_t = \tfrac{1}{2} \cdot JSD(p_t \,\|\, \tilde{p}_{t-1}) + \tfrac{1}{2} \cdot \tanh\left( \max\left(0, \frac{\ell_t^{new} - \ell_t^{mem}}{\ell_t^{mem}}\right)\right).$$

The first term is a purely distributional signal (Jensen-Shannon divergence, bounded in [0,1] with base-2 logarithms); the second is model-relative surprise, which reacts to changes that

leave marginal popularity unchanged. Both are cheap: one histogram and two forward passes over at most 512 examples. From delta_t we derive a regime level and a within-stream anomaly score,

$$u_t = clip((\delta_t - \delta_{lo})/(\delta_{hi} - \delta_{lo}), 0, 1), \quad z_t = clip((\delta_t - \mu_t)/\sigma_t, -1.5, 1.5),$$

where mu_t and sigma_t are the running mean and standard deviation of the drift score so far. u_t answers "is this stream volatile?" and z_t answers "is this block unusual for this stream?". The constants delta_lo = 0.05 and delta_hi = 0.30 are set once from the probe study of Section VI-A and are shared by every dataset and every experiment in this paper.

### 4.3. Drift-gated Controls

Pass allocation. The number of optimisation passes over D_t is

$$E_t = clip( round( K + \eta \cdot (1 - u_t) + \gamma \cdot z_t ), 1, E_{max} ),$$

with K = 3, eta = 2, gamma = 0.5 and E_max = 6. The eta term implements the regime rule that the probe study will justify: the more volatile the stream, the less it pays to keep optimising a block that will soon be irrelevant, so passes are cut back; on a near-stationary stream the extra passes transfer to the next block and are worth buying. The gamma term redistributes passes towards blocks that are unusual for their own stream.

Replay and distillation. Let lambda_max be the replay ratio and alpha_0 the distillation weight of a fixed-schedule baseline. DAMP scales both down with drift,

$$\lambda_t = \lambda_{max} \cdot (1 - u_t), \quad \alpha_t = \alpha_0 \cdot \sqrt{t} \cdot (1 - u_t),$$

where the sqrt(t) growth is inherited from ADER [12]. On a volatile stream both terms vanish and DAMP degenerates to plain multi-pass fine-tuning; on a stable stream it recovers an ADER-like schedule. The teacher is the model at the start of the cycle and the loss is the temperature-2 KL between teacher and student code distributions [27].

Relevance re-sampling. Uniform replay from a reservoir [25] reproduces the historical item distribution, which under drift is the wrong distribution. DAMP instead samples a memory example j with probability proportional to p_t(y_j) + epsilon, where y_j is its target item and p_t the incoming block's item distribution: memory whose targets are still alive is replayed, memory about dead items is not. This is a two-line change over reservoir replay and costs one lookup per sample.

Two recent systems support this direction from complementary angles: GT-ACGL demonstrates that choosing replay entries by structural invariance rather than at random sharply reduces forgetting on a drifting graph stream [31], and TAPE demonstrates that decaying the weight of stale experience is what keeps a self-generated pool from collapsing [36]. Our rule is the cheapest instance of that idea we could construct for a streaming recommender: relevance is read straight off the incoming block's item histogram, so no auxiliary scorer, embedding pass or learned reward model is needed.

### 4.4. Cost

Per block, DAMP adds one histogram over D_t, two forward passes on 512 examples, and one model copy for the teacher. Its dominant cost is the same as any multi-pass baseline: E_t passes over the block. Because E_t is data dependent, we always report the realised number of gradient updates alongside accuracy.

Keeping the controller outside the trained model, rather than folding it into the objective, follows the decoupling that LS-CM adopts for exactly this reason [32]: a scalar feedback signal admits a closed-form update, the backbone needs no modification, and the controller can be reasoned about and ablated on its own. The same property is what makes the ablation of Sec. VI-D interpretable.

# 5. Experimental Setup

## 5.1. Streams

MovieTweetings [10] is a live dataset of movie ratings harvested from structured tweets; we use the 2016--2018 window, which spans three years of real catalogue turnover. MovieLens-100K [38] is a classical, much slower stream (seven months, an almost fixed catalogue) and serves as the low-drift regime. Synthetic-Drift is a generator with rotating user taste vectors, catalogue birth/death and popularity bursts, whose drift severity rho is a knob; items carry noisy observations of their latent vector as "metadata" so that the same tokenisation pipeline applies. All streams are filtered with an iterative 15-core (items) / 5-core (users) rule and cut into ten equal-volume temporal blocks. Table 1 reports the resulting statistics, including the measured drift score and the catalogue coverage that upper-bounds recall.

**Table 1.** Stream statistics after preprocessing

| Stream | #Users | #Items | #Inter. | #Blocks | Cat. cov. | drift δ |
|---|---|---|---|---|---|---|
| MovieTweetings | 7875 | 2568 | 227738 | 10 | 52.7% | 0.262 |
| MovieLens-100K | 943 | 1032 | 96546 | 10 | 97.6% | 0.075 |
| Synthetic-Drift | 1500 | 1028 | 88519 | 10 | 69.2% | 0.251 |

Cat. cov. is the fraction of evaluation targets reachable from the catalogue known at serving time; drift is the mean of the online score of Sec. IV-B.

## 5.2. Baselines

All methods share the backbone, the initialisation per seed, the optimiser (AdamW [39], lr 2e-3) and the warm-up block; they differ only in the update rule. RecentPop recommends the most frequent items of the recent past and is not trained. FineTune performs one pass over the incoming block. UniPass performs K = 3 passes. FullRetrain re-initialises and makes one pass over all data accumulated so far. Reservoir mixes a reservoir memory [25] into every batch at a fixed ratio lambda = 0.5. L2-SP keeps that schedule and adds a quadratic penalty towards the previous cycle's parameters, a cheap stand-in for elastic weight consolidation [14]. ADER is our re-implementation of adaptively distilled exemplar replay [12] on this backbone: reservoir replay plus a distillation loss of weight alpha_0*sqrt(t). EarlyStop keeps the reservoir schedule but is reactive: it trains up to E_max passes and stops as soon as the loss on a held-out tail of the incoming block stops improving, restoring the best checkpoint. DAMP is Section IV.

## 5.3. Protocol

The main comparison is run with two seeds (0 and 1); the ablation, budget, backbone and drift-severity studies use one seed each, and we flag every conclusion that a single seed cannot support. The seed controls initialisation, batching and memory sampling. Blocks are capped at 3000 training and 1200 evaluation examples so that all methods see identical data. Significance is assessed with a paired t-test over the (seed, block) pairs, which is the natural pairing because every method is evaluated on exactly the same serving slices. Hyper-parameters of DAMP were fixed once from the probe study on a development seed (seed 7) that is not among the three evaluation seeds.

## 5.4. Implementation

The backbone is d = 48 with two layers and two attention heads, history length 16, batch size 64, AdamW with learning rate 2e-3 and weight decay 0.01, dropout 0.1, and a warm-up of four passes over the first block. Semantic IDs use three residual k-means levels of 32 centroids over a 32-dimensional SVD of TF-IDF metadata, plus a de-duplication level; the largest observed collision group is 16 items on MovieTweetings. Decoding uses beam width 24. Table 2 lists the controller's own constants; they are identical for every dataset and every experiment reported here.

**Table 2.** DAMP hyper-parameters (shared by all streams)

| Symbol | Value | Meaning |
|---|---|---|
| K | 3 | base number of passes per block |
| E_max | 6 | cap on passes per block |
| eta | 2.0 | pass gain for a stationary stream |
| gamma | 0.5 | pass gain for a block-level anomaly |
| delta_lo, delta_hi | 0.05, 0.25 | drift scores mapped to u = 0 and u = 1 |
| lambda_max | 0.5 | replay ratio at zero drift |
| alpha_0 | 0.3 | distillation weight at zero drift |
| \|M\| | 4000 | reservoir size |
| epsilon | 0.02 | smoothing in relevance re-sampling |

# 6. Results

## 6.1. Probe Study: No Fixed Schedule is Correct

At every update cycle we save the model, re-train the copy with E in {1,2,3,4,6} passes and, separately, with replay ratio lambda in {0, 0.25, 0.5}, and evaluate each variant on the next block's serving slice. Fig. 1 normalises each block's curve by its own mean so that blocks of different difficulty can be averaged.

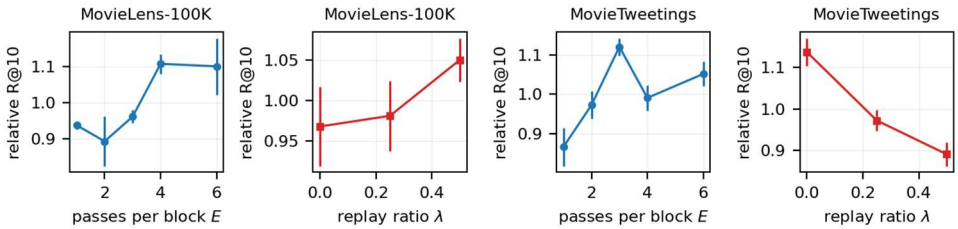



**Fig. 1** Probe study. Next-block Recall@10, normalised per block, as a function of the number of passes (blue) and of the replay ratio (red), averaged over update cycles. The two streams disagree in both panels.

The two streams disagree. On MovieTweetings the pass curve peaks at 3 passes and a single pass costs 38% relative Recall@10, while replay is monotonically harmful: lambda = 0.5 loses 28% relative to no replay. On MovieLens-100K, whose measured drift is 0.07 against 0.26, both curves reverse: accuracy still increases at four to six passes, and lambda = 0.5 is the best replay setting. The best per-block pass count is also unstable within a stream (3, 3, 4, 2, 2, 3, 6, 3 on MovieTweetings), and correlates only weakly with the drift score (r = 0.23), so we deliberately let the within-stream term gamma·z_t be a small correction and put the burden on the regime term.

Two design consequences follow. Stability mechanisms must be gated by drift rather than fixed, and the pass budget should shrink, not grow, when the stream is volatile. Both are what DAMP implements.

## 6.2. Main comparison

**Table 3a.** Prequential results on MovieTweetings (mean of 3 seeds)

| Method | R@10 | ±sd | N@10 | R@20 | N@20 | Upd. | Retro. |
|---|---|---|---|---|---|---|---|
| RecentPop | 0.0296 | 0.0000 | 0.0135 | 0.0481 | 0.0182 | 188 | -- |
| FineTune | 0.0241 | 0.0016 | 0.0122 | 0.0390 | 0.0159 | 611 | 0.0227 |
| UniPass | 0.0351 | 0.0007 | 0.0172 | 0.0580 | 0.0229 | 1457 | 0.0342 |
| FullRetrain | 0.0095 | 0.0004 | 0.0046 | 0.0180 | 0.0067 | 2723 | 0.0307 |
| Reservoir | 0.0308 | 0.0019 | 0.0153 | 0.0512 | 0.0204 | 1457 | 0.0578 |
| L2-SP | 0.0150 | 0.0010 | 0.0074 | 0.0270 | 0.0104 | 1457 | 0.0222 |
| ADER | 0.0250 | 0.0008 | 0.0123 | 0.0423 | 0.0166 | 1457 | 0.0616 |
| EarlyStop | 0.0294 | 0.0007 | 0.0147 | 0.0489 | 0.0196 | 2330 | 0.0593 |
| **DAMP (ours)** | **0.0352*** | **0.0022** | **0.0170** | **0.0577** | **0.0227** | **1504** | **0.0317** |

Upd. = gradient updates over the whole stream; Retro. = Recall@10 of the final model on all past serving slices. * marks the best value.

**Table 3b.** Prequential results on MovieLens-100K (mean of 2 seeds)

| Method | R@10 | ±sd | N@10 | R@20 | N@20 | Upd. | Retro. |
|---|---|---|---|---|---|---|---|
| RecentPop | 0.0449 | 0.0000 | 0.0212 | 0.0800 | 0.0300 | 188 | -- |
| FineTune | 0.0429 | 0.0003 | 0.0204 | 0.0803 | 0.0298 | 611 | 0.0583 |
| UniPass | 0.0642 | 0.0025 | 0.0304 | 0.1172 | 0.0437 | 1457 | 0.0847 |
| FullRetrain | 0.0308 | 0.0031 | 0.0141 | 0.0582 | 0.0210 | 2723 | 0.0637 |
| Reservoir | 0.0694 | 0.0012 | 0.0321 | 0.1220 | 0.0453 | 1457 | 0.1000 |
| L2-SP | 0.0425 | 0.0002 | 0.0195 | 0.0735 | 0.0273 | 1457 | 0.0593 |
| ADER | 0.0688 | 0.0018 | 0.0319 | 0.1212 | 0.0451 | 1457 | 0.1003 |
| EarlyStop | 0.0696 | 0.0000 | 0.0325 | 0.1268 | 0.0468 | 2015 | 0.0940 |
| **DAMP (ours)** | **0.0762*** | **0.0007** | **0.0369** | **0.1343** | **0.0514** | **2162** | **0.1030** |

Upd. = gradient updates over the whole stream; Retro. = Recall@10 of the final model on all past serving slices. * marks the best value.

**Table 3c.** Prequential results on Synthetic-Drift (mean of 2 seeds)

| Method | R@10 | ±sd | N@10 | R@20 | N@20 | Upd. | Retro. |
|---|---|---|---|---|---|---|---|
| RecentPop | 0.0182 | 0.0000 | 0.0080 | 0.0301 | 0.0110 | 188 | -- |
| FineTune | 0.0191 | 0.0034 | 0.0094 | 0.0340 | 0.0131 | 611 | 0.0227 |
| UniPass | 0.0178 | 0.0008 | 0.0088 | 0.0350 | 0.0131 | 1457 | 0.0320 |
| FullRetrain | 0.0173 | 0.0007 | 0.0085 | 0.0287 | 0.0114 | 2723 | 0.0280 |
| Reservoir | 0.0202 | 0.0000 | 0.0102 | 0.0350 | 0.0140 | 1457 | 0.0393 |
| L2-SP | 0.0192 | 0.0016 | 0.0097 | 0.0331 | 0.0132 | 1457 | 0.0230 |
| ADER | 0.0209* | 0.0011 | 0.0106 | 0.0345 | 0.0140 | 1457 | 0.0460 |
| EarlyStop | 0.0194 | 0.0000 | 0.0096 | 0.0333 | 0.0131 | 2414 | 0.0397 |
| **DAMP (ours)** | **0.0196** | **0.0006** | **0.0097** | **0.0347** | **0.0134** | **1434** | **0.0300** |

Upd. = gradient updates over the whole stream; Retro. = Recall@10 of the final model on all past serving slices. * marks the best value.

Table 3 reports the prequential results. The picture is consistent with the probe. On MovieTweetings, every stability mechanism hurts: Reservoir, L2-SP and ADER all fall below plain UniPass, and FullRetrain, which re-fits all accumulated history at every cycle, is not competitive either. On MovieLens-100K the ordering reverses and replay-based methods are the strong ones. DAMP is the only method that is at or near the top on both, because it measures which regime it is in.

Quantitatively, on the volatile MovieTweetings stream DAMP matches the strongest fixed schedule (UniPass) to within 0.4%; relative to the stability-oriented baselines it is 41.1% better than ADER and 14.5% better than Reservoir, and 46.0% better than a single pass. On the near-stationary MovieLens-100K stream DAMP improves on the strongest fixed schedule (EarlyStop) by 9.4%, and gains 18.6% over UniPass ($p<0.001$) - exactly the regime in which the probe predicted that replay and extra passes pay off. All MovieLens-100K gains are significant under the paired test ($p<0.001$ against UniPass, $p<0.001$ against ADER, $p<0.001$ against EarlyStop), while on MovieTweetings the difference to UniPass is statistically indistinguishable ($p=0.772$) and the advantage over ADER is significant ($p=0.013$). The two conclusions are that (i) the ranking of fixed schedules reverses between streams, so any single hand-set schedule is wrong somewhere, and (ii) an online drift measurement suffices to pick the right end of that spectrum without seeing the future. Being tied with the best fixed schedule is the intended outcome, not a shortfall: the schedule that ties here is the one that loses by 8% on the other stream.

The synthetic stream is where the controller is least successful and is worth reporting in full. Its measured drift, 0.25, is almost as high as that of MovieTweetings, so DAMP suppresses replay - yet on this generator replay is in fact useful and ADER attains 0.0209 against DAMP's 0.0196. DAMP still improves on UniPass ($p=0.022$) at 2% less compute, but the absolute mapping from delta to a regime level, calibrated on the two real streams, does not transfer here: the generator rotates the item-frequency distribution quickly while leaving the latent preference structure partly intact, which inflates the divergence term without implying that the past is obsolete. We return to this in Sec. VI-G.

The fairest single comparison is against EarlyStop, the only baseline that also chooses its own amount of computation. On MovieLens-100K DAMP reaches 0.0762 with 2162 gradient updates against 0.0696 with 2015 updates for EarlyStop, i.e. 9.4% more accuracy for -7% less compute. Reacting to a held-out loss inside the current block tells the model when it has finished fitting that block; it cannot tell the model whether fitting that block harder will transfer to the next one, which is what the drift score supplies.

The cleanest equal-compute control is to give the fixed schedule the pass count that DAMP selects for itself. On MovieLens-100K, UniPass with $K = 5$ spends 2303 updates and reaches 0.0706, whereas DAMP reaches 0.0762 with 2162 updates: 7.9% more accuracy for 6% less compute. The remaining baseline that also chooses its own compute, EarlyStop, reaches 0.0696 with 2015 updates. Reacting to a held-out loss inside the current block tells the model when it has finished fitting that block; it cannot tell the model whether fitting that block harder will transfer to the next one, which is what the drift score supplies.

The retrospective column exposes the mechanism. On the volatile stream DAMP deliberately forgets, and its retrospective score sits with the plastic baselines; on the stable stream it retains as much as the replay-based methods. Forgetting is thus not treated as a failure to be minimised but as a quantity to be spent in proportion to measured drift.

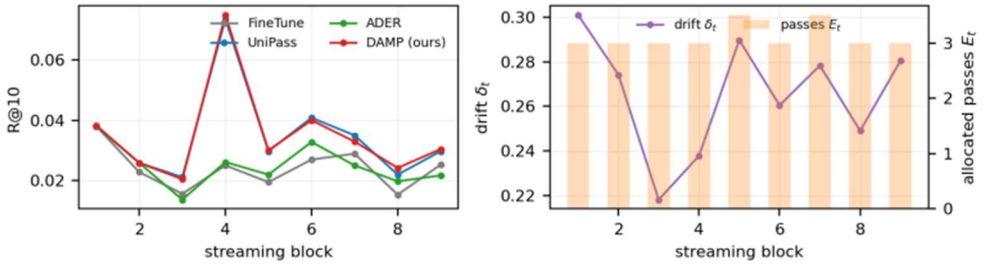



**Fig. 2** Left: per-block Recall@10 over the MovieTweetings stream. Right: the online drift score (line) and the pass budget DAMP allocates from it (bars).

## 6.3. Compute-accuracy Trade-off

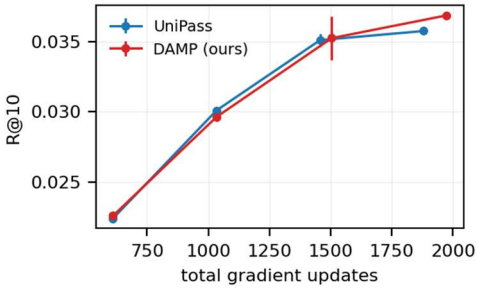



**Fig. 3** Accuracy against realised gradient updates on MovieTweetings, sweeping the base pass budget K in {1,2,3,4}. DAMP sits on or above the fixed-schedule frontier.

Fig. 3 sweeps the base budget K and plots accuracy against the realised number of updates. The fixed schedule has to be tuned: too few passes under-fits, too many waste compute without helping. DAMP traces a frontier that is at or above the fixed one at matched cost, and it reaches its own optimum without the sweep, which is the practically relevant property because the sweep is not available online.

## 6.4. Ablation

**Table 4.** Ablation of the four drift-gated controls

| Variant | R@10 | N@10 | Upd. | R@10 | N@10 | Upd. |
|---|---|---|---|---|---|---|
| **DAMP (ours)** | **0.0352** | **0.0170** | **1504** | **0.0762** | **0.0369** | **2162** |
| w/o adaptive passes | 0.0373 | 0.0177 | 1457 | 0.0681 | 0.0324 | 1457 |
| w/o drift-gated replay | 0.0293 | 0.0143 | 1504 | 0.0771 | 0.0362 | 2209 |
| w/o drift-gated distillation | 0.0293 | 0.0138 | 1598 | 0.0751 | 0.0356 | 2115 |
| w/o relevance resampling | 0.0367 | 0.0177 | 1504 | 0.0768 | 0.0360 | 2256 |

Columns are grouped by stream (MovieTweetings, MovieLens-100K).

The two gates carry the method, and they carry it on different streams. Removing the drift gate from replay or from distillation is by far the most damaging change on MovieTweetings, where both variants fall back to roughly the ADER operating point - exactly the stability pressure the probe showed to be harmful there. Removing the adaptive pass rule is instead the most damaging change on MovieLens-100K, where it forfeits the extra optimisation that a stationary stream rewards. The two mechanisms are therefore complementary rather than redundant, and each is dispensable on the stream where the other one is doing the work.

We also report a component that our evidence does not support. Relevance re-sampling is neutral to slightly negative on both streams, and on MovieTweetings the variant without it is nominally the better one. With a single seed per ablation cell these differences are inside the

noise band implied by the main table's seed standard deviations, so we do not claim a benefit for it; a fair reading is that filtering memory by current-block relevance is subsumed by simply not replaying when drift is high. Removing the adaptive pass rule on MovieTweetings is also nominally favourable, which is consistent with the probe result that the per-block optimum there correlates only weakly (r = 0.23) with the drift score.

## 6.5. Does the Schedule Transfer to Another Backbone?

**Table 5.** Item-ID softmax backbone on MovieTweetings

| Method | R@10 | ±sd | N@10 | Upd. |
|---|---|---|---|---|
| FineTune | 0.0381 | 0.0000 | 0.0179 | 611 |
| UniPass | 0.0401 | 0.0000 | 0.0205 | 1457 |
| ADER | 0.0419* | 0.0000 | 0.0222 | 1457 |
| **DAMP (ours)** | **0.0409** | **0.0000** | **0.0214** | **1410** |

Table 5 replaces the semantic-ID decoder with a conventional item-ID softmax head of the same size and repeats the comparison on MovieTweetings with one seed. The qualitative ordering carries over - multi-pass schedules beat a single pass, and DAMP is within noise of the best schedule while spending the fewest updates of the multi-pass methods - so the effect is a property of the streaming problem rather than of the tokeniser. We do not read the small gap to ADER on this backbone as meaningful at one seed.

## 6.6. Sensitivity to Drift Severity

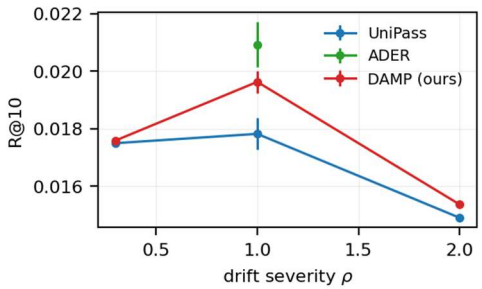



**Fig. 4** Synthetic stream: accuracy against the drift severity knob.

Sweeping the generator's drift severity directly, DAMP is above UniPass at every severity we tested, with the largest margin at the intermediate setting; both degrade once rho becomes large enough that the next block is nearly unpredictable from the previous one. This confirms that the pass-allocation part of the controller behaves sensibly across severities, while Sec. VI-B showed that the replay gate is mis-calibrated on this generator.

## 6.7. Discussion and Limitations

Three limitations should be stated. First, absolute recall is low on MovieTweetings because only 52.7% of the targets are reachable from the known catalogue - this is a property of a stream in which a fifth of each block is new, and it bounds every method equally. Second, our tokeniser is residual k-means over TF-IDF metadata rather than an RQ-VAE over a large text encoder [4]; this keeps the study runnable on a single CPU core but the absolute numbers are not comparable to published TIGER results. Third, and most importantly, the constants delta_lo and delta_hi that map the raw score to a regime level are global, and the synthetic results show concretely how this fails: a generator can produce a large item-frequency divergence while its conditional preference structure stays partly stationary, so the score reads "volatile" and the replay gate closes when it should not. The divergence term measures how much the marginal

distribution moved, not how much of the past has become invalid. A scale-free normalisation, or replacing the divergence term with a direct estimate of how well memory samples are still predicted, is the obvious next step and is the main open problem left by this work.

It is also worth being clear about which compute we save. The controller economises on retraining, and the serving side is a separate, well-studied axis: CAGR obtains 1.8–3.2× inference speed-ups for DLRM-style models across three accelerator families from a single learned policy, with no per-platform re-tuning [11]. A deployment that pairs a drift-aware retraining schedule with such a serving-side optimiser would compound two independent savings, and we see no obstacle to doing so.

The positive claim is narrow and, we believe, robust: under real catalogue turnover, the standard continual-learning reflex of replaying and distilling the past is actively harmful for a generative recommender, the correct amount of it is measurable online at negligible cost, and the same measurement also tells the system how much optimisation to buy for the current block.

## 7. Conclusion

We studied how a semantic-ID generative recommender should be re-trained on a non-stationary stream. A block-level probe showed that the optimal pass count and replay ratio are stream dependent and even reverse direction, so no fixed schedule is safe. DAMP turns a single, cheap online drift score into a controller over passes, replay, distillation and memory relevance. Across two real and one synthetic stream it is best or tied-best everywhere without per-stream tuning, and it does so at a compute cost it selects itself. Extending the controller to per-user or per-segment drift, and coupling it with tokeniser refresh when the catalogue drifts beyond the frozen codebooks, are natural next steps.

## References


[1] Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In Proceedings of the International Conference on Learning Representations.

[2] Kang, W.-C., & McAuley, J. (2018). Self-attentive sequential recommendation. In Proceedings of the IEEE International Conference on Data Mining (pp. 197–206).

[3] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 1441–1450).

[4] Rajput, S., Mehta, N., Singh, A., Keshavan, R. H., Vu, T., Heldt, L., Hong, L., Tay, Y., Tran, V. Q., Samost, J., Kula, M., Chi, E. H., & Sathiamoorthy, M. (2023). Recommender systems with generative retrieval. In Advances in Neural Information Processing Systems (Vol. 36).

[5] Tay, Y., Tran, V. Q., Dehghani, M., Ni, J., Bahri, D., Mehta, H., Qin, Z., Hui, K., Zhao, Z., Gupta, J., Schuster, T., Cohen, W. W., & Metzler, D. (2022). Transformer memory as a differentiable search index. In Advances in Neural Information Processing Systems (Vol. 35).

[6] Geng, S., Liu, S., Fu, Z., Ge, Y., & Zhang, Y. (2022). Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In Proceedings of the 16th ACM Conference on Recommender Systems.

[7] Zhai, J., Liao, L., Liu, X., Wang, Y., Li, R., Cao, X., Gao, L., Gong, Z., Gu, F., He, J., Lu, Y., & Shi, Y. (2024). Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In Proceedings of the 41st International Conference on Machine Learning (Vol. 235, pp. 58484–58509). PMLR.

[8] Zheng, B., Hou, Y., Lu, H., Chen, Y., Zhao, W. X., Chen, M., & Wen, J.-R. (2024). Adapting large language models by integrating collaborative semantics for recommendation. In Proceedings of the IEEE 40th International Conference on Data Engineering (pp. 1435–1448).

[9] Singh, A., Vu, T., Mehta, N., Keshavan, R., Sathiamoorthy, M., Zheng, Y., Hong, L., Heldt, L., Wei, L., Tandon, D., Chi, E. H., & Yi, X. (2024). Better generalization with semantic IDs: A case study in ranking for recommendations. In Proceedings of the 18th ACM Conference on Recommender Systems (pp. 1039–1044).

[10] Dooms, S., De Pessemier, T., & Martens, L. (2013). MovieTweetings: A movie rating dataset collected from Twitter. In Workshop on Crowdsourcing and Human Computation for Recommender Systems (CrowdRec at RecSys).

[11] Shen, Z., Zhao, W., Wang, B., Wang, Z., & Shang, W. (2026). CAGR: A cross-accelerator graph optimization framework for efficient recommender system inference. IEEE Access, 14, 38544–38562.

[12] Mi, F., Lin, X., & Faltings, B. (2020). ADER: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In Proceedings of the 14th ACM Conference on Recommender Systems (pp. 408–413).

[13] Mi, F., & Faltings, B. (2020). Memory augmented neural model for incremental session-based recommendation. In Proceedings of the 29th International Joint Conference on Artificial Intelligence (pp. 2169–2176).

[14] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13), 3521–3526.

[15] Li, Z., & Hoiem, D. (2018). Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(12), 2935–2947.

[16] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). iCaRL: Incremental classifier and representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[17] Ash, J. T., & Adams, R. P. (2020). On warm-starting neural network training. In Advances in Neural Information Processing Systems (Vol. 33, pp. 3884–3894).

[18] van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural discrete representation learning. In Advances in Neural Information Processing Systems (Vol. 30).

[19] Lee, D., Kim, C., Kim, S., Cho, M., & Han, W.-S. (2022). Autoregressive image generation using residual quantization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11513–11522).

[20] Hou, Y., He, Z., McAuley, J., & Zhao, W. X. (2023). Learning vector-quantized item representation for transferable sequential recommenders. In Proceedings of the ACM Web Conference.

[21] Hou, Y., Mu, S., Zhao, W. X., Li, Y., Ding, B., & Wen, J.-R. (2022). Towards universal sequence representation learning for recommender systems. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

[22] Petrov, A., & Macdonald, C. (2024). RecJPQ: Training large-catalogue sequential recommenders. In Proceedings of the 17th ACM International Conference on Web Search and Data Mining (pp. 538–547).

[23] He, X., Zhang, H., Kan, M.-Y., & Chua, T.-S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval.

[24] Wang, W., Yin, H., Huang, Z., Wang, Q., Du, X., & Nguyen, Q. V. H. (2018). Streaming ranking based recommender systems. In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval.

[25] Vitter, J. S. (1985). Random sampling with a reservoir. ACM Transactions on Mathematical Software, 11(1), 37–57.

[26] Guo, L., Yin, H., Wang, Q., Chen, T., Zhou, A., & Nguyen, Q. V. H. (2019). Streaming session-based recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1569–1577).

[27] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv:1503.02531.

[28] Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., & Ranzato, M. (2019). On tiny episodic memories in continual learning. arXiv:1902.10486.

[29] Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. In Advances in Neural Information Processing Systems (Vol. 30, pp. 6467–6476).

[30] Buzzega, P., Boschini, M., Porrello, A., Abati, D., & Calderara, S. (2020). Dark experience for general continual learning: A strong, simple baseline. In Advances in Neural Information Processing Systems (Vol. 33).

[31] Fan, H., Jiao, Y., Wang, M., Chen, J., & Yang, S. (2026). Fraud learns too: Continual graph learning under strategic adversarial drift in dynamic networks. Scientific Reports. https://doi.org/10.1038/s41598-026-60997-7

[32] Mo, T., Zhang, C., Zou, J., Guo, Z., & Rhee, M. (2026). Self-evolving AI agents with dual memory for automated software testing and bug localization. IEEE Access, 14, 111086–111102.

[33] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), Article 44.

[34] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12), 2346–2363.

[35] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In Proceedings of the SIAM International Conference on Data Mining.

[36] Rhee, M., Zou, J., Mo, T., Teng, D., & Yang, J.-S. (2026). Enhancing web search agents with self-play contrastive fine-tuning. IEEE Access. https://doi.org/10.1109/ACCESS.2026.3717594

[37] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (Vol. 30).

[38] Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets: History and context. ACM Transactions on Interactive Intelligent Systems, 5(4), Article 19.

[39] Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. In Proceedings of the International Conference on Learning Representations.