Intelligent & Human Futures
WISDOM ACADEMIC https://doi.org/10.63808/ihf.v2i3.472 Volume 2 , Issue 3

Research Article

A Hybrid Online-Offline Training Architecture for Maintaining Model Inference Freshness in Production Systems

Zhengyuan Wang[1,*]
[1]Johns Hopkins University, Baltimore 21218, United States
[*]**Correspondence:** Zhengyuan Wang, harleycodywzy94@gmail.com

Abstract

Production drift erodes predictive performance between scheduled rebuilds. Periodic retraining incurs cadence-bound freshness lag, whereas continuous online learning is exposed to transient noise. We formalise inference freshness as observable sample lag plus a predictive-loss proxy and present a hybrid controller that coordinates online updates, offline rebuilds, and shadow-validated swaps. On four non-stationary INSECTS streams, the hybrid improves prequential accuracy by 1.1–5.3 percentage points over the stronger of periodic-offline and online-only. ARF remains 0.7–16.3 points more accurate but processes 455–610 predictions/s versus 6,231–9,820 for the hybrid in this CPU replay. A corrupted-rebuild test rejects all poor candidates and avoids a 20.63-point next-1,000-prediction regression. On synthetic low-, medium-, and high-transition streams, periodic retraining leads by 1.6, 2.9, and 0.5 points. The contribution is therefore a transparent freshness-aware scheduling and update-safety design, not state-of-the-art predictive performance.

Keywords

Inference freshness, Concept drift adaptation, Online-offline hybrid training, Streaming machine learning, MLOps

1. Introduction

Production data drift can reduce accuracy before the next scheduled rebuild. Recent MLOps evidence identifies drift handling as a persistent operational problem, yet pipelines often accumulate ad hoc fixes instead of treating prediction freshness as an explicit requirement (Eken et al., 2025).

Calendar retraining and continuous online learning fail differently. Calendar schedules

ignore when distributions change, while incremental updates are sensitive to label noise, short regimes, and rollback or coordination problems. Streaming-drift reviews describe detection and adaptation algorithms but not their integration with the served state (Arora et al., 2024).

Production-AI studies commonly assess validation accuracy rather than freshness (Steidl et al., 2023). Continual-learning methods address forgetting but are usually evaluated on offline streams without delayed labels or constrained compute (Wang et al., 2024). How these mechanisms should coordinate to meet a measurable serving-freshness target remains unclear.

Three gaps follow: inference freshness lacks an operational metric; hybrid triggers and synchronisation remain heuristic; and comparisons under ordered, delayed-label workloads are scarce.

We address these gaps by defining freshness as observable lag plus an operational loss proxy; introducing a controller that triggers rebuilds and validates model handoffs; and comparing periodic-offline, online-only, ADWIN-retrain, ARF, and the hybrid in ordered replay. The hybrid improves on the three MLP alternatives, while ARF is more accurate but slower in this implementation.

The result is a tested architectural template for services with explicit prediction-freshness requirements.

2. Data and Methods

2.1. Problem Formulation and Freshness Metric

Consider an inference service issuing predictions over a non-stationary stream. The data's underlying distribution changes over time, becoming D_t at time t. The model serving the system at time t is f_t. At this point, $\tau(t)$ marks the latest timestamp where labeled samples have been added to f_t, through either an online update or an offline rebuild, as defined in section 2.3. The freshness lag, which we call $L(t)=t-\tau(t)$ and measure in samples, shows how out-of-date the model is compared to what it has recently seen. This delay is directly observable from the pipeline state. If a(t) is the wall-clock arrival timestamp of sample t, the corresponding time-based freshness age is $L_{time}(t)=a(t)-a(\tau(t))$; under a constant request rate q, this reduces to $\frac{L(t)}{q}$. The present replay reports sample lag because it does not impose an external request clock. The delay carries a structural floor at the label-arrival delay D: samples from the most recent D steps have not yet received labels and cannot have been absorbed. This temporal quantity is complemented by a freshness loss $\Delta L(t) = E_{D_t}[\ell(f_t)] - E_{D_t}[\ell(f_t^*)]$, the expected excess loss of f_t under D_t over the loss attainable by the hypothetical f_t^*

Intelligent & Human Futures
WISDOM ACADEMIC https://doi.org/10.63808/ihf.v2i3.472 Volume 2 , Issue 3

trained on D_t itself. Since the optimal up-to-date model is not observable, $\Delta L(t)$ is not directly estimated; instead, a rolling sequential error over a recent window is used as a proxy for it, and it is assumed that the loss attained by this optimal model is approximately stationary within the window. Treating $(L(t), \Delta L(t))$ jointly makes freshness a measurable scheduling goal rather than an indirect indication of accuracy, which is in line with accuracy-aware scheduling in feature store maintenance (Wooders et al., 2023).

Because the hypothetical current-optimal model is unobservable, $\Delta L(t)$ is not computed directly. Let $\mathcal{M}(t)$ denote the M = 1,000 most recent prequential prediction events. The controller uses the rolling error and its slow reference:

$$r(t)=\frac{1}{M}\sum_{s\in\mathcal{M}(t)} 1[\hat{y}_s\neq y_s], \quad b(t)=\beta b(t-1)+(1-\beta)r(t), \quad \beta=0.98 \tag{1}$$

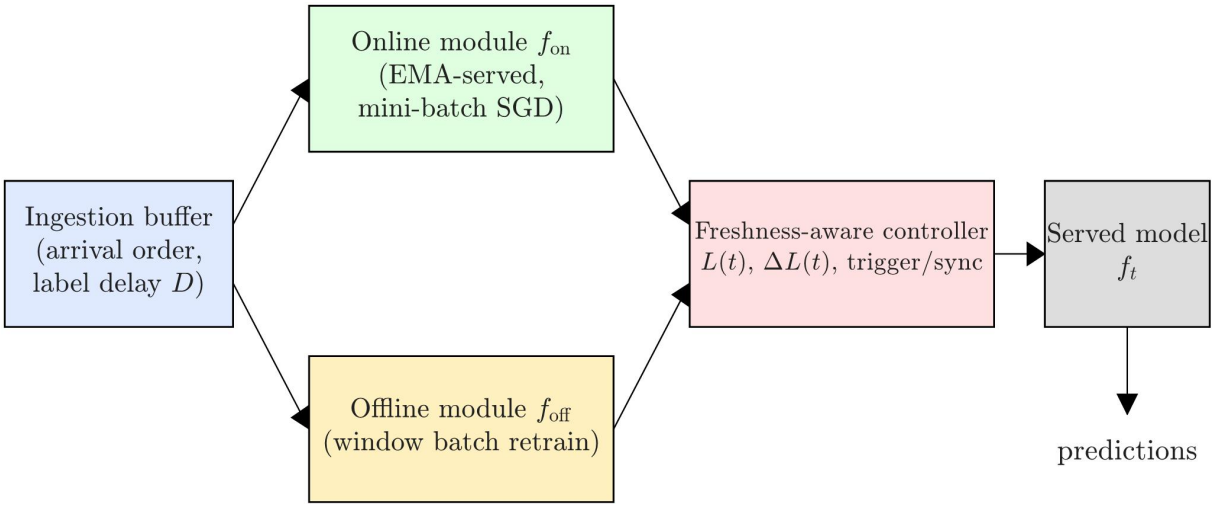
The baseline b is undefined until the rolling window first fills, is then initialised to r, and is updated after every mini-batch without being reset across rebuilds. The loss trigger fires when $r(t)>k\,b(t)$. This proxy assumes that the irreducible loss changes slowly over the rolling window and the EMA time scale; if that assumption fails, the ratio may reflect task difficulty as well as staleness.

2.2. System Architecture Overview

Figure 1 shows four components. The ingestion buffer preserves arrival order and releases labels after D samples. The online module performs one SGD step per labelled mini-batch and serves EMA-smoothed weights. The offline module rebuilds a base model on the latest W labels. The controller monitors lag and rolling loss, triggers rebuilds, and commits only shadow-validated candidates. The inference interface always exposes the last committed state.

Figure 1

Overall system architecture



2.3. Offline and Online Training Modules

The offline channel rebuilds an identical network from scratch on the latest W = 8,000 labelled samples using Adam (learning rate 10−3), cross-entropy, five epochs, and batch size 128. Reinitialisation removes parameter dependence on obsolete regimes, while W trades coverage against recency. Cost scales with both window and model size. Rebuilds take seconds for this MLP; larger backbones may require a budget-aware W or warm starts, with greater carry-over from old concepts. These are design options, not tested results.

The online channel starts from the last committed offline state and applies SGD (batch 32, momentum 0.9, learning rate 10−3, gradient clipping). EMA smoothing (decay 0.9) prevents raw updates from being served and keeps lag near its label-delay floor. This short-term tracking complements offline consolidation and addresses the instability of purely incremental adaptation under recurring drift (Suárez-Cetrulo et al., 2023).

Neither trainer decides when to refresh. Both expose weight loading, updating, rebuilding, and export operations. The controller can initialise the online path from an offline candidate, replay a held-out catch-up slice, atomically commit it, or abort without changing the served model.

2.4. Freshness-Aware Triggering and Model Synchronization

The controller monitors L(t) and r(t)/b(t). Each mini-batch updates the 1,000-sample prequential-loss window and its 0.98-decay baseline, then evaluates the rebuild trigger and, when triggered, the guarded handoff.

$$L(t)\geq L_{max} \vee r(t)>k\, b(t) \vee idle(t)\geq T_{idle} \quad (2)$$

As specified at line 7 of Algorithm 1, an offline rebuild is triggered when any of three conditions occurs: the freshness lag exceeds a safety limit, $L(t) \geq L_{max}$; the rolling prequential loss is more than k times its baseline, indicating that the online channel cannot keep up with the data drift; or the time since the last trigger has exceeded an idle limit, T_{idle}, thereby preventing endless inactivity even if there is slow drift without a sudden warning signal. To reduce transient noise, a small minimum spacing of h samples is used for all three. Because drift appears in several forms—abrupt, gradual, incremental, and recurring—the three triggers are complementary: relying on any single condition would leave some regimes undetected (Yang et al., 2026). In this hybrid structure, the online channel will keep L(t) close to its lower bound D, and L_{max} is therefore more of a safeguard than an upper limit; this is supported by Sections 3.4 and 4. The routinely adjusted parameter is the loss multiplier k; T_{idle} is used to handle slow drift as a backstop.

A simple swap of the new offline base into the serving slot would lose the online updates

that have occurred since the end of the offline window and thus reintroduce a delay at the time of refresh. This is avoided by rebuilding on a window that excludes the most recent labelled slice, validating the candidate on that held-out slice, and, only after acceptance, replaying the slice for catch-up before atomically swapping the served model and re-initialising the online module. The gate is evaluated before catch-up on the 500 most recent labelled samples held out of the rebuild window: if the accuracy of the new model on this slice drops by more than $\varepsilon = 0.02$ compared to that of the incumbent model, the swap is aborted, and the controller will wait for one hysteresis window before retrying. The slice is reported in samples rather than wall-clock because the sample-to-time mapping depends on the deployment's serving rate; 500 samples sits at half the controller's rolling-loss window, large enough for a stable accuracy estimate. Once the gate passes, the same held-out slice is replayed into the candidate through the online update rule. Triggering and synchronisation are based on freshness rather than calendar time. The loss multiplier k is the main operator-facing control; L_{max}, T_{idle}, h and ε act as safety bounds and are tuned once or infrequently adjusted.

$$A_{cand}+\varepsilon \geq A_{inc}, \quad \varepsilon=0.02 \tag{3}$$

A rejected candidate never changes the served state. The synchronous replay does not model crashes, buffer loss, or automatic restart; live deployment requires checkpointed recovery.

State alternates between SERVING and synchronous REBUILDING. COMMIT and ABORT both return to SERVING and reset hysteresis.

Algorithm 1

Freshness-aware controller (implementation order)

Input: stream S; delay D; rebuild window W; shadow slice V; rolling window M;
decay β; multiplier k; ceilings Lmax and Tidle; hysteresis h; tolerance ε.

Initialise fon and fserve on the warm-up window; set τ and lasttrigger.
For each arriving mini-batch [t,e):
    Predict with fserve; append prequential errors to the M-sample deque.
    When the deque is full, initialise or update b; compute r.
    Set c = e − D and update fon only on newly labelled samples below c.
    Update fserve by EMA and set τ = c.
    If hysteresis has elapsed and (L ≥ Lmax or r > kb or idle ≥ Tidle):
        Set g = c − V; train fcand from scratch on [g−W,g).
        Compare fcand and fserve on held-out slice [g,c).
        If Acand + ε ≥ Ainc: replay [g,c) into fcand; atomically COMMIT.
        Otherwise: ABORT and retain the incumbent served model.

12 Record rebuild cost and set lasttrigger = e on either exit.

2.5. Datasets, Baselines, Metrics, and Experimental Setup

Experiments use four class-balanced INSECTS streams from the USP DS Repository: abrupt (52,848 samples), gradual (24,150), incremental (57,018), and recurring (79,986), each with 33 features and six classes. River's Agrawal generator supplies 30,000-sample low-, medium-, and high-transition streams. Arrival order is preserved and labels are delayed by D = 200. Baselines are periodic-offline, rebuilding the MLP every 4,000 samples on W = 8,000; online-only, applying one SGD step per labelled batch (Gai et al., 2024); ADWIN-retrain, feeding delayed prequential errors to ADWIN (delta = 0.002; Bifet & Gavaldà, 2007) and rebuilding the same MLP for five epochs on the latest 8,000 labels; and ARF with 10 trees (Gomes et al., 2017). All methods use a 1,000-sample warm-up and three seeds. We report prequential accuracy, macro-F1, Cohen's κ, lag, median update latency, throughput, and cumulative rebuild time.

Base model: MLP with layer widths 33, 64, 64, 6 and ReLU activations. Online channel: SGD (momentum 0.9, learning rate 10^{-3}), gradient norm clipped at $\|g\|_2 \leq 1$, mini-batch B = 32, served-weight EMA decay α = 0.9. Offline channel: Adam (learning rate 10^{-3}), E = 5 epochs over window W = 8,000 in mini-batches of 128. Controller: rolling-loss window 1,000, baseline EMA decay β = 0.98, loss multiplier k = 1.5, lag ceiling L_{max} = 1,500, idle ceiling T_{idle} = 6,000, hysteresis h = 500, shadow tolerance ε = 0.02, warmup 1,000 samples. Component ablations and sensitivity sweeps are reported in section 3.4 and discussed in section 4.

3. Results

3.1. Inference Freshness vs. Baselines

The main comparison tests the hybrid against the two operating-mode baselines, ADWIN-triggered MLP retraining, and ARF on all four INSECTS variants. Table 1 reports accuracy, macro-F1, and Cohen's kappa for all five configurations; Table 2 then reports paired sequential uncertainty for the two decision-relevant comparisons.

Table 1

Main predictive comparison on the four INSECTS streams (mean ± SD over three seeds)

Variant	Configuration	Accuracy	Macro-F1	Cohen's κ
abrupt	periodic-offline	0.5261 ± 0.0018	0.5240 ± 0.0017	0.4314 ± 0.0022
abrupt	online-only	0.5860 ± 0.0004	0.5827 ± 0.0003	0.5033 ± 0.0005

abrupt	ADWIN-retrain	0.4499 ± 0.0037	0.4495 ± 0.0031	0.3398 ± 0.0044
abrupt	ARF	0.7127 ± 0.0007	0.7100 ± 0.0006	0.6553 ± 0.0009
abrupt	hybrid	0.5968 ± 0.0007	0.5939 ± 0.0012	0.5162 ± 0.0009
gradual	periodic-offline	0.5195 ± 0.0086	0.5075 ± 0.0112	0.4246 ± 0.0101
gradual	online-only	0.4894 ± 0.0057	0.4658 ± 0.0083	0.3876 ± 0.0068
gradual	ADWIN-retrain	0.3749 ± 0.0170	0.3584 ± 0.0201	0.2524 ± 0.0201
gradual	ARF	0.7350 ± 0.0034	0.7226 ± 0.0033	0.6813 ± 0.0041
gradual	hybrid	0.5724 ± 0.0012	0.5624 ± 0.0017	0.4869 ± 0.0015
incremental	periodic-offline	0.6036 ± 0.0021	0.6046 ± 0.0018	0.5243 ± 0.0026
incremental	online-only	0.5875 ± 0.0027	0.5737 ± 0.0039	0.5050 ± 0.0033
incremental	ADWIN-retrain	0.5517 ± 0.0048	0.5541 ± 0.0038	0.4621 ± 0.0058
incremental	ARF	0.6311 ± 0.0011	0.6242 ± 0.0006	0.5574 ± 0.0013
incremental	hybrid	0.6245 ± 0.0004	0.6206 ± 0.0006	0.5494 ± 0.0004
recurring	periodic-offline	0.5025 ± 0.0024	0.5066 ± 0.0018	0.4032 ± 0.0029
recurring	online-only	0.5954 ± 0.0016	0.5933 ± 0.0017	0.5144 ± 0.0019
recurring	ADWIN-retrain	0.5025 ± 0.0124	0.5085 ± 0.0120	0.4032 ± 0.0149
recurring	ARF	0.7494 ± 0.0014	0.7486 ± 0.0015	0.6993 ± 0.0017
recurring	hybrid	0.6123 ± 0.0063	0.6153 ± 0.0065	0.5348 ± 0.0075

As shown in Table 1, the hybrid exceeds the stronger of the two operating-mode baselines on every real stream by 1.1–5.3 percentage points in accuracy. The added comparators qualify that result: ADWIN-triggered retraining is weaker than the hybrid on these streams, whereas ARF is stronger by 0.7–16.3 percentage points. Thus, the evidence supports the controller as a scheduling alternative to the two MLP operating modes, but does not support a state-of-the-art predictive claim against adaptive ensembles.

Because correctness observations are serially dependent, paired circular moving-block bootstrap was used instead of an independent-sample test (block length 1,000; 2,000 resamples). Table 2 reports the mean paired accuracy difference, 95% interval, two-sided bootstrap p value, and standardized block effect (the mean non-overlapping block difference divided by its across-block standard deviation). Full results against all four comparators are supplied with the reproducibility package.

Table 2

Paired block-bootstrap comparisons on the four INSECTS streams

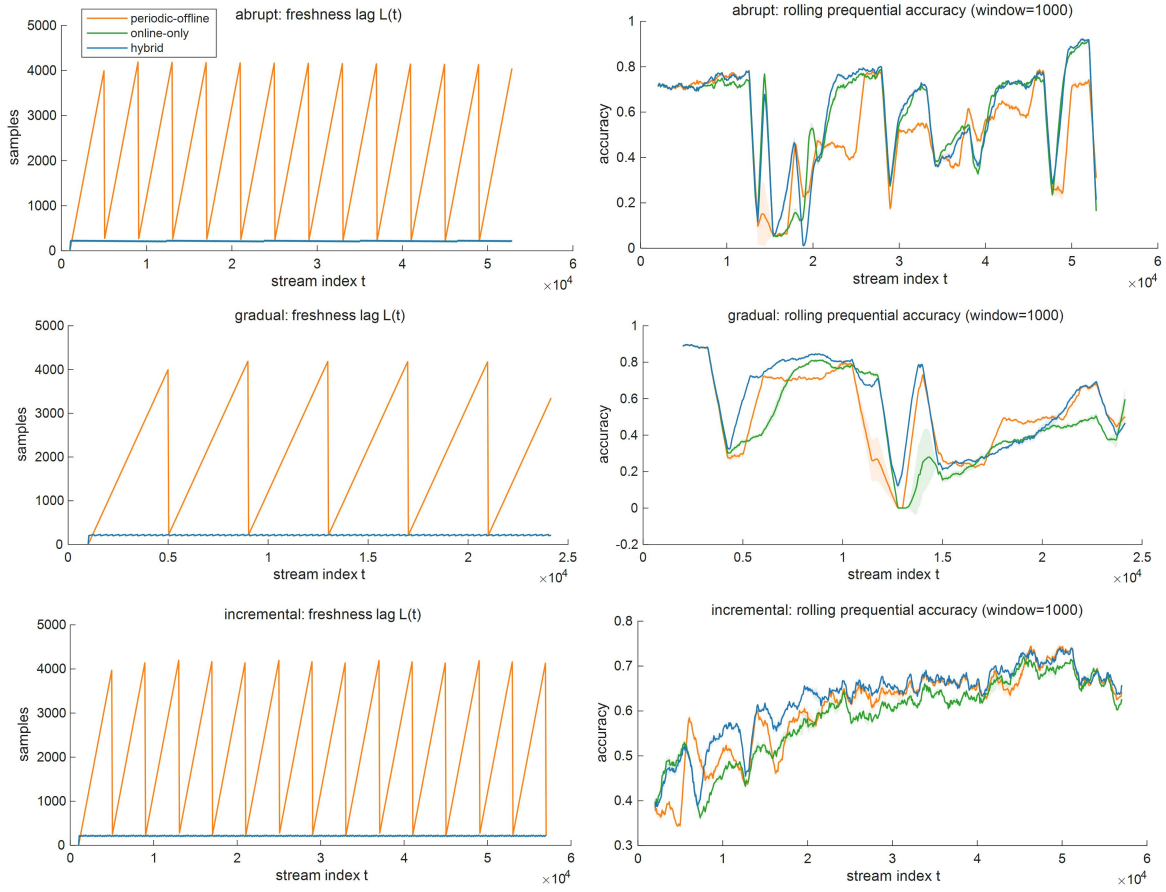
Intelligent & Human Futures
WISDOM ACADEMIC https://doi.org/10.63808/ihf.v2i3.472 Volume 2 , Issue 3

Variant	Comparison	Δ acc. (pp)	95% CI (pp)	p	Block effect
abrupt	hybrid − online-only	+1.07	[+0.02, +2.13]	0.0460	+0.151
abrupt	hybrid − ARF	-11.59	[-14.65, -8.59]	0.0010	-0.572
gradual	hybrid − periodic-offline	+5.29	[+2.70, +8.28]	0.0010	+0.426
gradual	hybrid − ARF	-16.26	[-20.67, -12.30]	0.0010	-0.836
incremental	hybrid − periodic-offline	+2.09	[+1.56, +2.62]	0.0010	+0.513
incremental	hybrid − ARF	-0.66	[-1.35, -0.03]	0.0390	-0.152
recurring	hybrid − online-only	+1.69	[+0.41, +3.02]	0.0110	+0.161
recurring	hybrid − ARF	-13.71	[-15.91, -11.56]	0.0010	-0.781

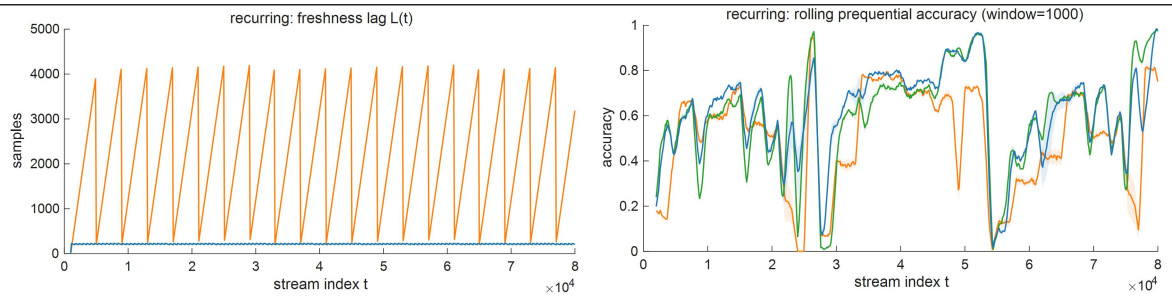
The total margins in Table 1 do not show when the three configurations deviate from each other. Freshness lag L(t) and rolling prequential accuracy (window 1,000) are tracked for the four streams; Figure 2 reports the mean and standard deviation over three seeds.

Figure 2

Freshness lag and rolling prequential accuracy over time on the four INSECTS variants (mean ± SD over three seeds)



Intelligent & Human Futures
WISDOM ACADEMIC https://doi.org/10.63808/ihf.v2i3.472 Volume 2 , Issue 3



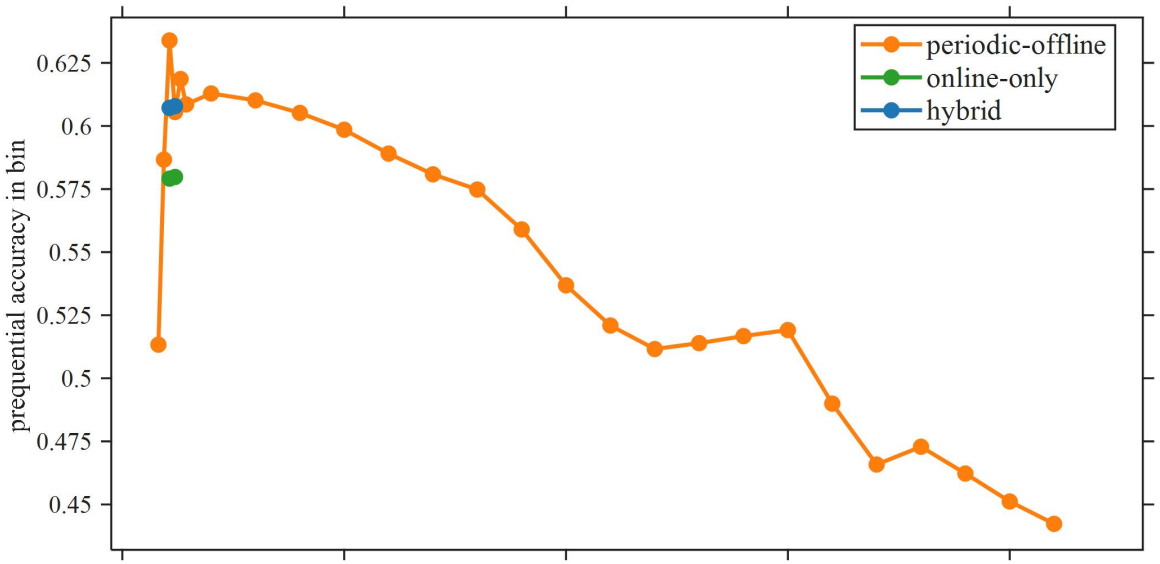
As shown in Figure 2, the three configurations are in different operating modes. Periodic-offline has the typical textbook sawtooth lag profile bounded by its rebuild interval; the accuracy declines as lag accumulates between rebuilds and is only partially recovered upon rebuilding, whereas online-only and hybrid both suffer from lag at the structural lower bound of $L \approx 215$ samples; the hybrid trace shows the highest aggregate accuracy on the four real streams, although the curves cross locally as regimes change.

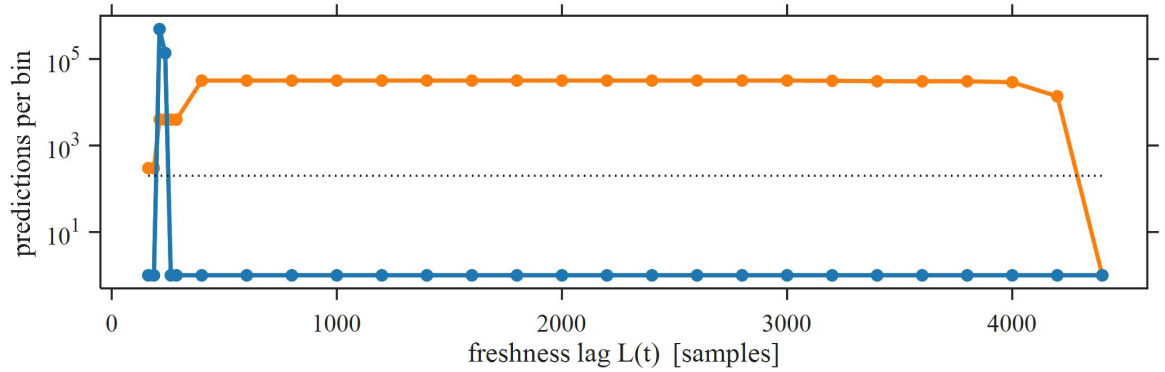
3.2. Effect of Freshness on Predictive Accuracy

A natural objection to the comparison in section 3.1 is that it is circular: the hybrid is both fresher and more accurate, so its accuracy advantage may be a tautological result of its lag advantage rather than evidence that a lower lag leads to higher accuracy. To break this circularity, prediction events from the four INSECTS streams are pooled across the three configurations and binned according to the freshness lag at the time of inference; then, the mean of the prequential accuracy per bin is calculated, and each configuration contributes predictions to the bins it actually reaches. The shared grid is intended to support matched-staleness comparisons, but such comparisons require more than one configuration to populate the same bin.

Figure 3

Prequential accuracy and prediction counts on a shared freshness-lag grid





As shown in Figure 3, periodic-offline accuracy generally declines as lag increases, with local non-monotonicity. The slope is steepest in the high-lag tail where periodic-offline frequently appears, and only 2 of 27 bins contain enough predictions from more than one configuration for comparison. The stratification supports a within-configuration decline of periodic-offline accuracy with lag, but does not isolate freshness from inductive bias across configurations. Cohen's κ was used as the main discriminator in all the strata of drift here because there was a class imbalance among the drift segments under investigation (Chen et al., 2024).

3.3. Latency, Throughput, and Retraining Cost

In addition to the predictive quality, the hybrid architecture also needs to have high-throughput serving performance under dual-training-channel operation. The runs that produced Table 1 are re-examined based on the three system-level axes measured directly from the implemented pipeline: sustained throughput, median online update latency, and cumulative offline retraining wall-clock time. The periodic-offline, online-only, ADWIN-retrain, and hybrid configurations use the same MLP and serving interface, so their system differences mainly reflect scheduling. ARF has different model capacity and update mechanics. Parameter count, FLOPs, and retraining budget are not equalised across ARF and the MLP methods; ARF therefore serves as a predictive reference, while its throughput is an implementation-specific measurement rather than an equal-capacity or equal-compute comparison.

All configurations ran on CPU only on a workstation with a six-core Intel Core i5-10400F processor and 16 GB of RAM under 64-bit Windows 10, using Python 3.14.3, PyTorch 2.12.0 (CPU build), scikit-learn 1.9.0, River 0.25.0, and NumPy 2.4.6. PyTorch used six intra-operation threads, and peak resident memory reached 429.6 MB. Throughput is the number of served predictions divided by total streaming-loop wall-clock; synchronous rebuilds, including rejected rebuilds, are included in that denominator and in cumulative retraining cost.

The hybrid sustained 6,231–9,820 predictions per second across the four variants, lower than 25,015–28,597 for online-only and 11,111–13,066 for periodic-offline. ADWIN-retrain sustained 6,952–19,584 predictions per second, whereas the 10-tree ARF sustained 455–610. These figures come from a shared replay harness and are not claimed to transfer directly to a deployed serving cluster; the ARF figure additionally reflects a different model family. The median online update delay of all incremental-update configurations is below one millisecond. The hybrid's cumulative offline retraining wall-clock ranges from 2.5 s in gradual mode to 7.0 s in recurring mode, corresponding to 31.9% below to 60.7% above periodic-offline depending on the variant.

3.4. Robustness under Drift and Ablation Study

The above comparisons used the four INSECTS variants as a real but uncurated stress test. The two extensions are as follows. An ablation study re-runs the recurring INSECTS variant with the online channel, the offline channel, or the synchronization handoff individually disabled, and the full hybrid is used as a reference row to attribute the hybrid's gains to specific architectural components. An extra set of streams is generated by the Agrawal concept-drift family in River, producing low, medium, and high regimes that vary the count and abruptness of injected concept transitions over a fixed 30,000-sample horizon, and the three configurations in Table 1 are evaluated in each regime. Tables 3 and 4 present the two analyses according to the design pattern proposed by Palli et al. (2025) for nonstationary imbalanced streams.

Table 3

Component ablation on the recurring INSECTS stream

Configuration	Accuracy	Macro-F1	Cohen's κ	Mean lag	Retrains
Full hybrid	0.612 ± 0.006	0.615 ± 0.007	0.535 ± 0.008	215	20.3
Without online	0.613 ± 0.004	0.616 ± 0.004	0.536 ± 0.004	893	68.7
Without offline	0.570 ± 0.001	0.568 ± 0.001	0.484 ± 0.002	215	0.0
Without handoff	0.607 ± 0.003	0.608 ± 0.002	0.528 ± 0.003	220	20.3

Table 4

Predictive performance under synthetic concept drift

Regime	Configuration	Accuracy	Macro-F1	Cohen's κ	Mean lag	Retrains
Low	periodic-offline	0.805 ± 0.004	0.801 ± 0.004	0.605 ± 0.009	2120	7.0
Low	online-only	0.655 ± 0.007	0.651 ± 0.007	0.305 ± 0.013	215	0.0

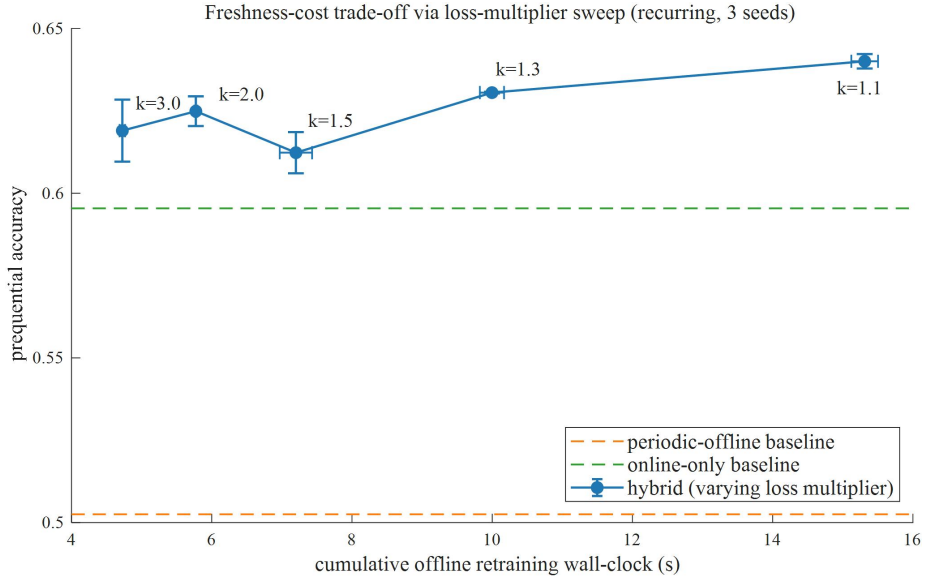
Low	hybrid	0.789 ± 0.002	0.786 ± 0.002	0.573 ± 0.003	215	11.0
Medium	periodic-offline	0.711 ± 0.002	0.710 ± 0.002	0.423 ± 0.004	2120	7.0
Medium	online-only	0.603 ± 0.004	0.600 ± 0.003	0.204 ± 0.008	215	0.0
Medium	hybrid	0.682 ± 0.003	0.681 ± 0.003	0.365 ± 0.007	215	7.0
High	periodic-offline	0.701 ± 0.007	0.699 ± 0.007	0.401 ± 0.015	2120	7.0
High	online-only	0.593 ± 0.012	0.592 ± 0.012	0.186 ± 0.025	215	0.0
High	hybrid	0.695 ± 0.005	0.695 ± 0.004	0.390 ± 0.009	215	6.3

As shown in Table 3, the ablation describes component-level trade-offs; Table 4 identifies the synthetic regimes in which calendar-based retraining remains superior. Across the ordinary recurring-stream hybrid runs, the controller proposed 20.3 swaps, accepted 13.7, and rejected 6.7 on average over three seeds. Excluding the online path increases mean lag from 215 to 893 samples and raises the number of offline retraining cycles from 20.3 to 68.7; eliminating the offline path reduces Cohen's κ from 0.535 to 0.484; and omitting the synchronization handoff reduces accuracy from 0.612 to 0.607 and raises mean lag from 215 to 220 samples. On the synthetic Agrawal streams, periodic-offline leads the hybrid by 1.6, 2.9, and 0.5 percentage points in the low-, medium-, and high-transition regimes, respectively; a strong base model is produced by periodic retraining over a structured Agrawal concept, and the online channel of the hybrid adds adaptive cost without proportional gain.

A sensitivity analysis on the recurring variant traces the cost-freshness trade-off as a function of the loss multiplier k, with other hyperparameters fixed at the section 2.5 values.

Figure 4

Predictive performance and cumulative retraining cost as a function of the loss multiplier k (markers and whiskers show mean ± SD over three seeds)



As shown in Figure 4, rebuild frequency and cumulative retraining time decline with k, whereas predictive accuracy changes non-monotonically. Across the five settings, the largest between-seed SD is 0.94 percentage points for accuracy and 0.23 s for cumulative retraining.

Reducing k from 1.5 to 1.1 increases accuracy by 2.8 percentage points while increasing offline wall-clock by about 113%; increasing k to 3.0 saves about 34% of that wall-clock, while accuracy is 0.7 percentage points above the k = 1.5 result, confirming that the predictive response is non-monotonic. A parallel sweep over $L_{max} \in \{500, 1500, 3000, 6000\}$ produced the same output; so the lag ceiling is non-binding over the evaluated range, while k controls rebuild frequency under the operating conditions of this study.

To test the shadow gate directly, the third proposed recurring-stream rebuild was deliberately corrupted by permuting its 8,000-window training labels. Each seed was run twice: once with the gate enforced and once with that injected candidate forcibly accepted; all other logic was unchanged. The candidate's shadow accuracy was on average 0.367 below the incumbent. The gate rejected all 3 corrupted proposals and preserved mean accuracy of 0.529 over the following 1,000 predictions, versus 0.322 under forced acceptance—an avoided regression of 20.63 percentage points.

Table 5

Controlled poor-rebuild test on the recurring stream (three seeds)

Condition	Poor model accepted	Next-1,000 accuracy	Samples to next accepted rebuild
Gate enforced	0/3	0.529	11403
Injected gate bypass	3/3	0.322	512

Note. The longer interval to the next accepted rebuild in the gated branch is not a recovery delay: service continued with the incumbent and no immediate recovery was required. In the bypass branch, the corrupted model reduced near-term accuracy and triggered a subsequent accepted rebuild after about 512 samples.

4. Discussion

The empirical pattern in section 3 supports the structural separation introduced in section 2 within the small-MLP setting: the online channel limits short-horizon freshness lag, while the offline channel provides long-horizon consolidation. The hybrid outperforms the two operating-mode baselines and ADWIN-triggered MLP retraining, but ARF is more accurate on all four streams; this result narrows the contribution from a broad concept-drift claim to a transparent freshness-aware scheduling design with a different accuracy-throughput operating

point. The ordinary ablation shows only a modest handoff effect, whereas the controlled corrupted-rebuild test validates its safety role directly: all three poor candidates were rejected and a 20.63-point next-1,000-prediction regression was avoided.

The sensitivity sweep in Figure 4 lifts the cost-freshness trade-off from a deployment-specific observation to a property of the controller family, and the loss-multiplier threshold k is an operator-facing cost-control knob: rebuild frequency and cumulative retraining time decline as k increases, whereas predictive accuracy is non-monotonic. The cost response is monotonic, but the predictive response is not; the sweep therefore supports k as a cost-control parameter rather than a guarantee of monotonic accuracy improvement. Practically speaking, any deployment of the proposed system selects its operating point on a single curve indexed by k, and L_{max} and T_{idle} act as safety bounds rather than regular tuning knobs.

The service owner defines the acceptable freshness-cost policy, while the operator implements it through k and the safety bounds. A rejected swap is an observable quality-control event: the candidate completed training but did not match the incumbent within ε on held-out data. The research implementation records proposal, acceptance, rejection, and shadow-accuracy statistics; a production deployment would additionally need durable model identifiers, trigger reasons, window cutoffs, and rollback records for auditability.

The evaluation has several limitations. It replays a static archive rather than a live deployment and thus does not reproduce variable request rates, missing labels, resource contention, or genuinely asynchronous training. The INSECTS domain is simpler than large commercial workloads, and the synthetic Agrawal streams sometimes favour full-window periodic retraining. The base learner is a small MLP whose rebuilds complete within seconds, so catch-up pressure for larger backbones remains untested. ARF is one representative adaptive ensemble rather than an exhaustive streaming benchmark, and the comparison is not compute-matched across model families. The reported results therefore establish behavior in the evaluated CPU replay, not state-of-the-art stream-learning performance or production-scale guarantees.

Future work should evaluate variable label delays and request rates in a live asynchronous service, repeat the study with larger backbones whose rebuild and catch-up phases are material, and compare a wider set of adaptive ensembles under equal-compute or equal-retraining budgets. Those experiments would determine whether the present controller settings transfer beyond the replay harness and whether its transparent freshness-cost policy remains useful when model capacity and resource contention dominate.

5. Conclusion

This paper introduces a hybrid online-offline architecture that makes inference freshness an explicit scheduling objective. Across four real INSECTS streams, it improves accuracy by 1.1–5.3 percentage points over the stronger of periodic-offline and online-only and keeps mean lag near the D = 200 structural bound. The stronger ARF baseline is 0.7–16.3 points more accurate, however, so the contribution is an accuracy-throughput and governance-oriented scheduling alternative rather than a state-of-the-art predictor. The controlled poor-rebuild experiment further shows that shadow validation can prevent a measurable near-term regression.

The main limits are the static replay rather than live serving, a small MLP whose rebuilds finish within seconds, and the absence of equal-compute comparisons across model families. The results therefore apply to the evaluated CPU replay and require validation with variable delays, larger models, and resource contention.

The contribution thus elevates freshness from an indirect symptom monitored after the fact into a tractable scheduling target indexable by a single operator-facing knob, providing a candidate template for production systems whose performance requirements include explicit requirements concerning how recently the served model has been updated, pending validation under live serving conditions. Beyond the specific empirical setting examined here, the formal decomposition of staleness and the freshness-aware coordination mechanism offer a general design pattern that can be adapted to other streaming workloads in which the cost of computational refresh must be balanced against the value of timely predictions. This perspective points toward a path forward for principled freshness management across a broader class of production machine-learning systems.

Acknowledgments

The author declares that no specific acknowledgments are required.

Author contributions

Zhengyuan Wang: Conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, and visualization. The author has read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data availability

The INSECTS datasets used in this study are publicly available from the USP DS Repository (Souza et al., 2020; https://sites.google.com/view/uspdsrepository). The synthetic streams were generated with River using the configurations and random seeds reported in Section 2.5. No proprietary or restricted datasets were used.

Declarations

Conflict of interest

The author declares that there is no conflict of interest.

Ethics approval

Not applicable.

Consent to participate

Not applicable.

Consent for publication

The author has given consent for publication.

Additional information

Received: 13 July 2026

Accepted: 3 August 2026

Published online: 19 August 2026

Publisher's Note

References

[1] Arora, S., Rani, R., & Saxena, N. (2024). A systematic review on detection and adaptation of concept drift in streaming data using machine learning techniques. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 14(4), e1536. https://doi.org/10.1002/widm.1536

[2] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. Proceedings of the 2007 SIAM International Conference on Data Mining, 443-448. https://doi.org/10.1137/1.9781611972771.42

[3] Chen, Y., Yang, X., & Dai, H. L. (2024). Cost-sensitive continuous ensemble kernel learning for imbalanced data streams with concept drift. Knowledge-Based Systems, 284, 111272. https://doi.org/10.1016/j.knosys.2023.111272

[4] Eken, B., Pallewatta, S., Tran, N., Tosun, A., & Babar, M. A. (2025). A multivocal review of MLOps practices, challenges and open issues. ACM Computing Surveys, 58(2), 1-35. https://doi.org/10.1145/3747346

[5] Gai, Y., Meng, K., & Wang, X. (2024). Online learning for streaming data classification in nonstationary environments. Statistical Analysis and Data Mining: The ASA Data Science Journal, 17(2), e11669. https://doi.org/10.1002/sam.11669

[6] Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfahringer, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. Machine Learning, 106(9-10), 1469-1495. https://doi.org/10.1007/s10994-017-5642-8

[7] Palli, A. S., Jaafar, J., Md Saad, M. H., Mokhtar, A. A., Gomes, H. M., Soomro, A. A., & Gilal, A. R. (2025). Smart adaptive ensemble model for multiclass imbalanced nonstationary data streams. Scientific Reports, 15(1), 21140. https://doi.org/10.1038/s41598-025-05122-w

[8] Souza, V. M. A., Reis, D. M. D., Maletzke, A. G., & Batista, G. E. A. P. A. (2020). Challenges in benchmarking stream learning algorithms with real-world data. Data Mining and Knowledge Discovery, 34(6), 1805-1858. https://doi.org/10.1007/s10618-020-00698-5

[9] Steidl, M., Felderer, M., & Ramler, R. (2023). The pipeline for the continuous development of artificial intelligence models: Current state of research and practice. Journal of Systems and Software, 199, 111615. https://doi.org/10.1016/j.jss.2023.111615

[10] Suárez-Cetrulo, A. L., Quintana, D., & Cervantes, A. (2023). A survey on machine learning for recurring concept drifting data streams. Expert Systems with Applications, 213, 118934. https://doi.org/10.1016/j.eswa.2022.118934

[11] Wang, L., Zhang, X., Su, H., & Zhu, J. (2024). A comprehensive survey of continual learning: Theory, method and application. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(8), 5362-5383. https://doi.org/10.1109/TPAMI.2024.3367329

[12] Wooders, S., Mo, X., Narang, A., Lin, K., Stoica, I., Hellerstein, J. M., Crooks, N., & Gonzalez, J. E. (2023). RALF: Accuracy-aware scheduling for feature store maintenance. Proceedings of the VLDB Endowment, 17(3), 563-576. https://doi.org/10.14778/3632093.3632116

[13] Yang, S., Han, M., Zhu, S., Yang, W., Dai, Z., Li, J., & Ding, J. (2026). A survey of processing methods for different types of concept drift. Data & Knowledge Engineering, 161, 102525. https://doi.org/10.1016/j.datak.2025.102525